SiFive U54-MC Core Complex Manual
v1p0

SiFive U54-MC Core Complex Manual

Proprietary Notice

Release Information

Version	Date	Changes
v1p0	October 4th, 2017	• Initial Release

Contents

Chapter 1

Overview

SiFive's U54-MC Core Complex is a high-performance full-Linux-capable cache-coherent 64-bit RISC-V multiprocessors available as an IP block. The U54-MC Core Complex follows all applicable RISC-V standards, and this document should be read in conjunction with the official RISC-V user-level and privileged-architecture standard documents.

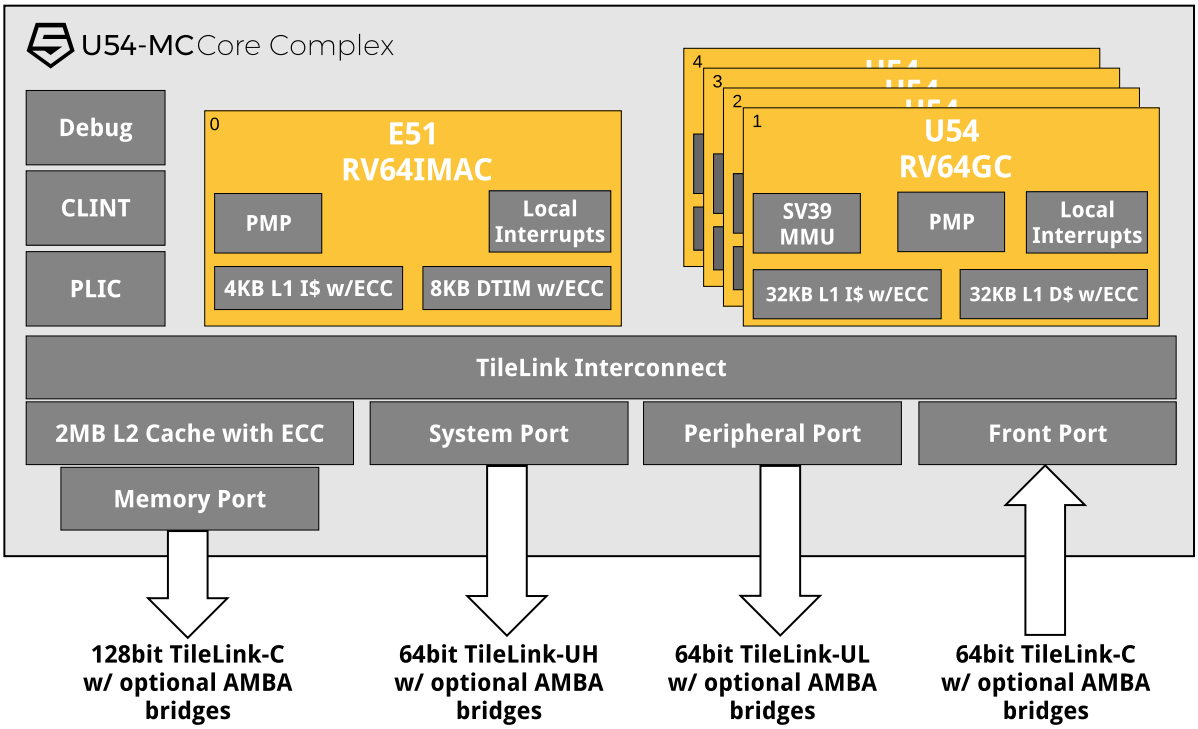

Figure 1.1: U54-MC Core Complex Block Diagram

The U54-MC Core Complex is shown in Figure 1.1. The U54-MC Core Complex includes 4x 64-bit RISC-V U54 cores with private caches and a shared L2 cache, a 64-bit RISC-V E51 monitor core, a platform-level interrupt controller (PLIC), a core local interruptor (CLINT), a JTAG-accessible debug

unit, a memory port for high-bandwidth memory access, and separate ports for I/O accesses. The L2 cache within the U54-MC Core Complex acts as the coherence hub in a system, and provides an external caching master port to support external high-bandwidth cache-coherent masters.

A summary of features in the U54-MC Core Complex can be found in Table 1.1.

U54-MC Core Complex Feature Set	
Feature	**Description**
Number of cores	5 cores.
RISC-V Core Name	4x U54 RISC-V cores.
Monitor Core	E51 RISC-V monitor core.
Local Interrupts	48 Local Interrupt signals per hart which can be connected to off core complex devices.
Level-2 Cache	2 MiB, 16-way, L2 Cache.
PLIC Interrupts	Support for 511 interrupts. 3 interrupts are from on core complex peripherals, the remaining 508 interrupts can be connected to the PLIC from off core complex devices.
PLIC Priority Levels	7 PLIC priority levels.
Hardware Breakpoints	2 hardware breakpoints.
Physical Memory Protection Unit	Each hart in the RISC-V Core IP has a PMP with 8x regions and a minimum granularity of 4 bytes.

Table 1.1: U54-MC Core Complex Feature Set

1.1 U54 RISC-V Application Cores

The U54-MC Core Complex is configured with 4x 64-bit cache-coherent U54 RISC-V application cores. Each U54 core has a high-performance single-issue in-order 64-bit execution pipeline, with a peak sustained execution rate of one instruction per clock cycle. U54 cores support a comprehensive dynamic branch prediction scheme, including BTB, BHT, and return-address stacks exploiting local and global history to improve performance. The cores support the standard RV64IMAFDC ISA, including full hardware support for single and double-precision IEEE 754-2008 floating-point with fully pipelined fused multiply-adds, a hardware divide and square-root unit, and full hardware support for subnormal numbers. A hardware integer multiplier and divider is also provided. The U54 core supports the standard C compressed extension for reduced code size. The U54 cores implement up to 512 GiB of virtual address space using the Sv39 virtual address translation scheme with a hardware page-table walker for address-translation cache refills.

The U54 RISC-V core is described in more detail in Chapter 3.

1.2 E51 RISC-V Monitor Core

The U54-MC Core Complex includes an E51 RISC-V monitor core to securely boot the system and service low-level interrupts and other tasks without disturbing the larger cores. The E51 supports the RV64IMAC ISA. The monitor core has full coherent access to the shared memory system and all peripherals. The monitor core can continue to service lightweight tasks while the application cores and L2 cache are in deep sleep to save power.

The E51 RISC-V core is described in more detail in Chapter 4.

1.3 Interrupts

The U54-MC Core Complex supports 48 high-priority, low-latency local vectored interrupts per-hart. This RISC-V Core IP includes a RISC-V standard platform-level interrupt controller (PLIC), which supports 511 global interrupts with 7 priority levels. This RISC-V Core IP also provides the standard RISC-V machine-mode timer and software interrupts via the Core Local Interruptor (CLINT).

Interrupts are described in Chapter 7, the PLIC in Chapter 8, and the CLINT in Chapter 9.

1.4 Memory System

Each U54 core's private L1 instruction and data caches are configured to be a 8-way set-associative 32 KiB caches.

The E51 monitor core has an 2-way set-associative 4 KiB L1 instruction cache.

All on-chip memory structures are protected with parity and/or ECC and all cores in the RISC-V Core IP have Physical Memory Protection (PMP) units.

The Level 1 memories are described in Chapter 3, the PMP is described in Chapter 10, and the L2 Cache Controller is described in Chapter 11.

1.5 Debug Support

The U54-MC Core Complex includes extensive platform-level debug facilities including hardware breakpoints, watchpoints, and single-step execution accessed via an industry-standard JTAG interface and supported by a full set of open-source debug tools. All components in the system, including processors, accelerators, memories, peripheral devices, and the interrupt controller, can be controlled and monitored over the debug port.

Debug support is described in detail in Chapter 12 and the debug interface is described in Chapter 13.

1.6 External Bus Interfaces

The U54-MC Core Complex has System, Memory, and Peripheral Ports for accessing off core complex address space. The System Port conform to the TL-UH specification and can be used to access high-speed off core complex devices. The Memory Port supports cacheable transactions via the TL-C specification and is connected to main memory. The Peripheral Port supports the TL-UL specification and is typically connected to lower speed peripherals.

There is also a TL-C bus interface, called the Front Port, which allows off core complex masters to access on core complex devices, such as the data and instruction tightly integrated memories.

More details on the U54-MC Core Complex ports can be found in Chapter 5 and Chapter 6.

Chapter 2

Terminology

CLINT	Core Local Interruptor. Generates per-hart software interrupts and timer interrupts.
Hart	HARdware Thread
DTIM	Data Tightly Integrated Memory
ITIM	Instruction Tightly Integrated Memory
JTAG	Joint Test Action Group
LIM	Loosely Integrated Memory. Used to describe memory space delivered in a SiFive Core Complex but not tightly integrated to a CPU core.
PMP	Physical Memory Protection
PLIC	Platform-Level Interrupt Controller. The global interrupt controller in a RISC-V system.
TileLink	A free and open interconnect standard originally developed at UC Berkeley.
RO	Used to describe a Read Only register field.
RW	Used to describe a Read/Write register field.
WO	Used to describe a Write Only registers field.
WARL	Write-Any Read-Legal field. A register field that can be written with any value, but returns only supported values when read.
WIRI	Writes-Ignored, Reads-Ignore field. A read-only register field reserved for future use. Writes to the field are ignored, and reads should ignore the value returned.
WLRL	Write-Legal, Read-Legal field. A register field that should only be written with legal values and that only returns legal value if last written with a legal value.
WPRI	Writes-Preserve Reads-Ignore field. A register field that may contain unknown information. Reads should ignore the value returned, but writes to the whole register should preserve the original value.

Chapter 3

U54 RISC-V Core

This chapter describes the 64-bit U54 RISC-V processor core used in the U54-MC Core Complex. The processor core comprises an instruction memory system, an instruction fetch unit, an execution pipeline, a data memory system, and support for local interrupts.

The U54 feature set is summarized in Table 3.1.

U54 Feature Set	
Feature	**Description**
ISA	RV64IMAFDC.
Instruction Cache	32 KiB 8-way instruction cache.
Instruction Tightly Integrated Memory	The U54 has support for an ITIM with a maximum size of 28 KiB.
Data Cache	32 KiB 8-way data cache.
Virtual Memory Support	The U54 has support for Sv39 virtual memory support with a 39-bit virtual address space, 38-bit physical address space, and a 32 entry TLB.
Modes	The U54 supports the following modes: Machine Mode, Supervisor Mode, User Mode.

Table 3.1: U54 Feature Set

3.1 Instruction Memory System

The instruction memory system consists of a dedicated 32 KiB 8-way set-associative instruction cache. The access latency of all blocks in the instruction memory system is one clock cycle. The instruction cache is not kept coherent with the rest of the platform memory system. Writes to instruction memory must be synchronized with the instruction fetch stream by executing a FENCE.I instruction.

The instruction cache has a line size of 64 B and a cache line fill will trigger a burst access outside of the U54-MC Core Complex. The core will cache instructions from executable addresses, with the exception of the ITIM, which is further described in Section 3.1.1. Please see the U54-MC Core Complex Memory Map in Chapter 6 for a description of executable address regions which are denoted by the attribute X.

Trying to execute an instruction from a non-executable address will result in a synchronous trap.

3.1.1 I-Cache Reconfigurability

The instruction cache can be partially reconfigured into an Instruction Tightly Integrated Memory (ITIM), which occupies a fixed address range in the memory map. ITIM provides high-performance, predictable instruction delivery. Fetching an instruction from ITIM is as fast as an instruction-cache hit, with no possibility of a cache miss. ITIM can hold data as well as instructions, though loads and stores to ITIM are not as performant as loads and stores to DTIM.

The instruction cache can be configured as ITIM for all ways except for 1 in units of cache lines (64 B bytes). A single instruction cache way must remain an instruction cache. ITIM is allocated simply by storing to it. A store to the nth byte of the ITIM memory map reallocates the first $n + 1$ bytes of instruction cache as ITIM, rounded up to the next cache line.

ITIM is deallocated by storing zero to the first byte after the ITIM region, i.e. 28 KiB after the base address of ITIM as indicated in the Memory Map in Chapter 6. The deallocated ITIM space is automatically returned to the instruction cache.

For determinism, software must clear the contents of ITIM after allocating it. It is unpredictable whether ITIM contents are preserved between deallocation and allocation.

3.2 Instruction Fetch Unit

The U54 instruction fetch unit contains branch prediction hardware to improve performance of the processor core. The branch predictor comprises a 40-entry branch target buffer (BTB) which predicts the target of taken branches, a 128-entry branch history table (BHT), which predicts the direction of conditional branches, and a 2-entry return-address stack (RAS) which predicts the target of procedure returns. The branch predictor has a one-cycle latency, so that correctly predicted control-flow instructions result in no penalty. Mispredicted control-flow instructions incur a three-cycle penalty.

The U54 implements the standard Compressed (C) extension to the RISC-V architecture which allows for 16-bit RISC-V instructions.

3.3 Execution Pipeline

The U54 execution unit is a single-issue, in-order pipeline. The pipeline comprises five stages: instruction fetch, instruction decode and register fetch, execute, data memory access, and register writeback.

The pipeline has a peak execution rate of one instruction per clock cycle, and is fully bypassed so that most instructions have a one-cycle result latency. There are several exceptions:

- LW has a two-cycle result latency, assuming a cache hit.
- LH, LHU, LB, and LBU have a three-cycle result latency, assuming a cache hit.
- CSR reads have a three-cycle result latency.
- MUL, MULH, MULHU, and MULHSU have a 5-cycle result latency.

- DIV, DIVU, REM, and REMU have between a 2-cycle and 33-cycle result latency, depending on the operand values.

The pipeline only interlocks on read-after-write and write-after-write hazards, so instructions may be scheduled to avoid stalls.

The U54 implements the standard Multiply (M) extension to the RISC-V architecture for integer multiplication and division. The U54 has a 16-bit per cycle hardware multiply and a 4-bit per cycle hardware divide.

Branch and jump instructions transfer control from the memory access pipeline stage. Correctly-predicted branches and jumps incur no penalty, whereas mispredicted branches and jumps incur a three-cycle penalty.

Most CSR writes result in a pipeline flush with a five-cycle penalty.

3.4 Data Memory System

The U54 data memory system has a 8-way set-associative 32 KiB write-back data cache that supports 64 B cache lines. The access latency is two clock cycles for words and double-words, and three clock cycles for smaller quantities. Misaligned accesses are not supported in hardware and result in a trap to support software emulation. The data caches are kept coherent with a directory-based cache coherence manager, which resides in the outer L2 cache.

Stores are pipelined and commit on cycles where the data memory system is otherwise idle. Loads to addresses currently in the store pipeline result in a five-cycle penalty.

3.5 Atomic Memory Operations

The U54 core supports the RISC-V standard Atomic (A) extension on the DTIM and the Peripheral Port. Atomic memory operations to regions that do not support them generate an access exception precisely at the core.

The load-reserved and store-conditional instructions are only supported on cached regions, hence generate an access exception on DTIM and other uncached memory regions.

See The RISC-V Instruction Set Manual, Volume I: User-Level ISA, Version 2.1 [1] for more information on the instructions added by this extension.

3.6 Floating-Point Unit (FPU)

The U54 FPU provides full hardware support for the IEEE 754-2008 floating-point standard for 32-bit single-precision and 64-bit double-precision arithmetic. The FPU includes a fully pipelined fused-multiply-add unit and an iterative divide and square-root unit, magnitude comparators, and float-to-integer conversion units, all with full hardware support for subnormals and all IEEE default values.

3.7 Virtual Memory Support

The U54 has support for virtual memory through the use of a Memory Management Unit (MMU). The MMU supports the Bare and Sv39 modes as described in The RISC-V Instruction Set Manual, Volume II: Privileged Architecture, Version 1.10 [2].

The U54 MMU has a 39-bit virtual address space mapped to a 38-bit physical address space. A hardware page-table walker refills the address translation caches. Both instruction and data address translation caches are fully associative, and have 32 entires. The MMU supports 2 MiB megapages and 1 GiB gigapages to reduce translation overheads for large contiguous regions of virtual and physical address space.

Note that the U54 does not automatically set the *Accessed* (A) and *Dirty* (D) bits in a Sv39 Page Table Entry (PTE). Instead, the U54 MMU will raise a page fault exception for a read to a page with PTE.A=0 or a write to a page with PTE.D=0.

3.8 Local Interrupts

Each U54 supports up to 48 local interrupt sources that are routed directly to the core. See Chapter 7 for a detailed description of Local Interrupts.

3.9 Supported Modes

The U54 supports RISC-V supervisor and user modes, providing three levels of privilege: machine (M), supervisor (S) and user (U).

Supervisor mode adds a number of additional CSRs and capabilities. Please see The RISC-V Instruction Set Manual, Volume II: Privileged Architecture, Version 1.10 [2] for more information on the supported privilege modes.

3.10 Physical Memory Protection (PMP)

The U54-MC Core Complex includes a Physical Memory Protection Unit compliant with The RISC-V Instruction Set Manual, Volume II: Privileged Architecture, Version 1.10 [2]. PMP can be used to set memory access privileges (read, write, execute) for specified memory regions. The U54-MC Core Complex PMP supports 8 regions with a minimum region size of 4 bytes.

See Chapter 10 for more information on the PMP.

3.11 Hardware Performance Monitor

The U54-MC Core Complex supports a basic hardware performance monitoring facility compliant with The RISC-V Instruction Set Manual, Volume II: Privileged Architecture, Version 1.10 [2]. The `mcycle` CSR holds a count of the number of clock cycles the hart has executed since some arbitrary time in the past. The `minstret` CSR holds a count of the number of instructions the hart has retired since some arbitrary time in the past. Both are 64-bit counters. The hardware performance monitor includes two additional event counters, `mhpmcounter3` and `mhpmcounter4`. The event selector CSRs `mhpmevent3` and `mhpmevent4` are registers that control which event causes the corresponding counter to increment. The `mhpmcounter`s are 40-bit counters.

The event selectors are partitioned into two fields, as shown in Table 3.2: the lower 8 bits select an event class, and the upper bits form a mask of events in that class. The counter increments if the event corresponding to any set mask bit occurs. For example, if `mhpmevent3` is set to `0x4200`, then `mhpmcounter3` will increment when either a load instruction or a conditional branch instruction retires. Note, an event selector of 0 means “count nothing.”

Machine Hardware Performance Monitor Event Register	
Instruction Commit Events, `mhpeventX`[7:0] = 0	
Bits	**Meaning**
8	Exception taken
9	Integer load instruction retired
10	Integer store instruction retired
11	Atomic memory operation retired
12	System instruction retired
13	Integer arithmetic instruction retired
14	Conditional branch retired
15	JAL instruction retired
16	JALR instruction retired
17	Integer multiplication instruction retired
18	Integer division instruction retired
19	Floating-point load instruction retired
20	Floating-point store instruction retired
21	Floating-point addition retired
22	Floating-point multiplication retired
23	Floating-point fused multiply-add retired
24	Floating-point division or square-root retired
25	Other floating-point instruction retired
Microarchitectural Events, `mhpeventX`[7:0] = 1	
Bits	**Meaning**
8	Load-use interlock
9	Long-latency interlock
10	CSR read interlock
11	Instruction cache/ITIM busy
12	Data cache/DTIM busy
13	Branch direction misprediction
14	Branch/jump target misprediction
15	Pipeline flush from CSR write
16	Pipeline flush from other event
17	Integer multiplication interlock
18	Floating-point interlock
Memory System Events, `mhpeventX`[7:0] = 2	
Bits	**Meaning**
8	Instruction cache miss
9	Data cache miss or memory-mapped I/O access
10	Data cache writeback
11	Instruction TLB miss
12	Data TLB miss

Table 3.2: `mhpmevent` Register Description

Chapter 4

E51 RISC-V Monitor Core

The E51 monitor core is a 64-bit embedded RISC-V microcontroller, including an instruction fetch unit, an execution pipeline, and a data memory system. The monitor core supports the standard RISC-V RV64IMAC user-level instruction set, with machine and user privilege modes.

4.1 E51 Instruction Fetch Unit

The E51 instruction fetch unit consists of 2-way set-associative 4 KiB instruction cache that supports 64 B cache lines. The access latency is one clock cycle.

The instruction memory system is not coherent with the data memory system. Writes to memory may be synchronized with the instruction fetch stream with a FENCE.I instruction.

The branch predictor comprises a branch target buffer (BTB), which predicts the target of taken branches and jumps; a branch history table (BHT), which predicts the direction of conditional branches; and a return-address stack (RAS), which predicts the target of procedure returns. The BTB is configured to hold 40 entries. The RAS is configured to hold 2 entries. The BHT uses a `gshare` prediction scheme with 7 bits of global history to access an array of 128 two-bit saturating counters. The branch predictor has a one-cycle latency, so that correctly predicted control-flow instructions result in no penalty.

4.2 E51 Execution Pipeline

The E51 execution unit is a single-issue, in-order pipeline. The pipeline comprises five stages: instruction fetch, instruction decode and register fetch, execute, data memory access, and register writeback.

The pipeline has a peak execution rate of one instruction per clock cycle. It is fully bypassed, so that most instructions have an apparent one-cycle result latency. There are several exceptions:

- LD and LW have a two-cycle result latency, assuming a cache hit.
- LH, LHU, LB, and LBU have a three-cycle result latency, assuming a cache hit.
- MUL, MULW, MULH, MULHU, MULHSU, DIV, DIVU, REM, REMU, DIVW, DIVUW, REMW, and REMUW have between a 2-cycle and 66-cycle result latency, depending on operand values.

- CSR reads have a three-cycle result latency.

The pipeline only interlocks on read-after-write and write-after-write hazards, so instructions may be scheduled to avoid stalls.

The iterative multiplier is configured to produce 16-bits per cycle with an early-out option. The iterative divider has latency of between three and 66 cycles and an early-out option.

Branch and jump instructions transfer control from the memory access pipeline stage. Correctly-predicted branches and jumps incur no penalty, whereas mispredicted branches and jumps incur a three-cycle penalty.

Most CSR writes result in a pipeline flush, a five-cycle penalty.

4.3 E51 Data Memory System

The E51 data memory system consists of a 8 KiB tightly-integrated RAM (TIM). The access latency is two clock cycles for full words and three clock cycles for smaller quantities. Misaligned accesses are not supported in hardware and result in a trap to support software emulation.

Stores are pipelined and commit on cycles where the data memory system is otherwise idle. Loads to addresses currently in the store pipeline result in a five-cycle penalty.

Chapter 5

U54-MC Core Complex Interfaces

This chapter describes the primary interfaces to the U54-MC Core Complex.

5.1 Clock & Reset

The `core_clock`, `rtc_toggle`, `clock`, `reset`, and `reset_vector` inputs are described in Table 5.1.

The relationship between the clock input frequencies are as follows: `core_clock` $\geq$ `clock` $>$ 2$\times$ `rtc_toggle`

Name	Direction	Width	Description
`core_clock`	Input	1	The core pipeline and cache clock.
`clock`	Input	1	Clock input to the PLIC, Clint, Debug, and the external ports. Has a $1/m$ frequency relationship with `core_clock` where $m \geq 1$.
`rtc_toggle`	Input	1	The Real Time Clock input. Must run at strictly less than half the rate of `clock`.
`reset`	Input	1	Synchronous reset signal. Active high. Must be asserted for 16 cycles of `clock` and synchronously deasserted.
`reset_vector`	Input	38-bit	Reset Vector Address. Implementations MUST set this signal to a valid address.

Table 5.1: Clock and Reset Interfaces

5.1.1 Real Time Clock (`rtcToggle`)

As defined in the RISC-V privileged specification, RISC-V implementations must expose a real-time counter via the `mtime` register. In the U54-MC Core Complex the `rtcToggle` input is used as the real-time counter. `rtcToggle` must run at strictly less than half the frequency of `clock`. Furthermore, for RISC-V compliance, the frequency of `rtcToggle` must remain constant, and software must be made aware of this frequency.

5.1.2 System Clock (`clock`)

The system clock is used to decouple the frequency of the core from that of some of the on core complex peripherals. `clock` has a $1/m$ frequency relationship with `core_clock` where m is any positive integer. Additionally, these clocks must be phase-aligned.

The peripherals connected to clock are: PLIC, CLINT, Debug, and the ports.

5.2 Ports

This section will describe all of the Ports in the U54-MC Core Complex.

5.2.1 Front Port

The U54-MC Core Complex has a master bus interface called Front Port. This port can be used by external masters to read and write into the local U54-MC Core Complex DTIM and to the ITIM address space. Note that an external master using the Front Port can trigger the I-Cache to reconfigure itself by using the procedure described in Section 3.1.1.

Reads and writes to the Font Port interface can also pass through to the other RISC-V Core IP ports if a transaction falls within their address space. Transactions through the Front Port do not pass through a PMP.

The Front Port is described in Table 5.2.

5.2.2 Memory Port

The U54-MC Core Complex Memory Port is used to access off core complex cacheable address space, typically main memory. The System Port is mapped into two address ranges; one in the lower 32-bits of the address map, and one above 32-bits.

The Memory Port is described in Table 5.2.

5.2.3 System Port

The U54-MC Core Complex System Port is used to access off core complex un-cached address space typically for accessing higher bandwidth peripherals. The System Port is mapped into two address ranges; one in the lower 32-bits of the address map, and one above 32-bits.

The System Port is described in Table 5.2.

5.2.4 Peripheral Port

The U54-MC Core Complex Peripheral Port is used to access off core complex address space, typically peripheral devices. The System Port is mapped into two address ranges; one in the lower 32-bits of the address map, and one above 32-bits.

The Peripheral Port is described in Table 5.2.

Name	Base Address	Top	Protocol	Description
front_port_tl_0	NA	NA	TL-C	64-bit data width. Master Bus interface allows for other bus masters to access U54-MC Core Complex address space.
periph_port_tl_0	0x2000_0000 0x01_0000_0000	0x4FFF_FFFF 0x0F_FFFF_FFFF	TL-UL	64-bit data width. Synchronous to clock.
system_port_tl_0	0x5000_0000 0x10_0000_0000	0x7FFF_FFFF 0x1F_FFFF_FFFF	TL-UH	64-bit data width. Synchronous to clock.
memory_port_tl_0	0x8000_0000 0x20_0000_0000	0xFFFF_FFFF 0x3F_FFFF_FFFF	TL-C	Cacheable. 128-bit data width. Synchronous to clock.

Table 5.2: U54-MC Core Complex Ports

5.3 Local Interrupts

Local interrupts are interrupts which can be connected to peripheral sources and signaled directly to an individual hart. Please see Chapter 7 for a detailed description of the U54-MC Core Complex local interrupts.

Name	Direction	Width	Description
local_interrupts_0	Input	48	E51 Local Interrupts. These are level-based interrupt signals connected directly to the core and must be synchronous with core_clock.
local_interrupts_1	Input	48	U54 Hart 1 Local Interrupts. These are level-based interrupt signals connected directly to the core and must be synchronous with core_clock.
local_interrupts_2	Input	48	U54 Hart 2 Local Interrupts. These are level-based interrupt signals connected directly to the core and must be synchronous with core_clock.
local_interrupts_3	Input	48	U54 Hart 3 Local Interrupts. These are level-based interrupt signals connected directly to the core and must be synchronous with core_clock.
local_interrupts_4	Input	48	U54 Hart 4 Local Interrupts. These are level-based interrupt signals connected directly to the core and must be synchronous with core_clock.

Table 5.3: Local Interrupt Interface

5.4 Global Interrupts

The global_interrupt signals are inputs into the U54-MC Core Complex PLIC. Note that there are 511 total Global Interrupts, however some on core complex devices are pre-connected to the PLIC, such as the L2 Cache Controller. Only 508 global_interrupt signals are exposed at the top level.

Name	Direction	Width	Description
global_interrupts	Input	508	External interrupts from off-chip or peripheral sources. These are level-based interrupt signals connected to the PLIC and must be synchronous with clock.

Table 5.4: External Interrupt Interface

5.5 Unrecoverable Errors

The unrecoverable error signals are generated by the ECC logic when a multi-bit error is detected. All unrecoverable error signals are positive level triggered and remain high until reset.

Name	Direction	Width	Description
halt_from_tile_0	Output	1	Signals an unrecoverable error has occurred in hart 0
halt_from_tile_1	Output	1	Signals an unrecoverable error has occurred in hart 1
halt_from_tile_2	Output	1	Signals an unrecoverable error has occurred in hart 2
halt_from_tile_3	Output	1	Signals an unrecoverable error has occurred in hart 3
halt_from_tile_4	Output	1	Signals an unrecoverable error has occurred in hart 4
halt_from_l2_0	Output	1	Signals an unrecoverable error has occurred in the L2 Cache

Table 5.5: Unrecoverable Error Signals

5.6 Test Mode Reset Interface

The test mode reset interface provides a mechanism for the internal RISC-V Core IP debug reset to be driven directly. Note that this has no relationship to general RISC-V Core IP 'reset'.

Name	Direction	Width	Description
debug_psd_test_mode	Input	1	When asserted, reset synchronization logic in the RISC-V Core IP is bypassed
debug_psd_test_mode_reset	Input	1	This signal resets debug logic in the RISC-V Core IP when debug_psd_test_mode is also asserted

Table 5.6: Test Mode Reset Interface

5.7 Debug Output Signals

Signals which are outputs from the Debug Module are shown in Table 5.7.

Name	Direction	Width	Description
ndreset	Output	1	This signal is a reset signal driven by the Debug Logic of the chip. It can be used to reset parts of the SoC or the entire chip. It should NOT be wired into logic which feeds back into the jtag_reset signal for this block. This signal may be left unconnected
dmactive	Output	1	This signal, 0 at reset, indicates that debug logic is active. This may be used to prevent power gating of debug logic, etc. It may be left unconnected

Table 5.7: External Debug Logic Control Pins

5.8 JTAG Debug Interface Pinout

SiFive uses the industry-standard JTAG interface which includes the four standard signals, TCK, TMS, TDI, and TDO. A test logic reset signal must also be driven on the jtag_reset input. This reset is synchronized internally to the design. The test logic reset must be pulsed before the core reset is deasserted.

Name	Direction	Width	Description
jtag_TCK	Input	1	JTAG Test Clock
jtag_TMS	Input	1	JTAG Test Mode Select
jtag_TDI	Input	1	JTAG Test Data Input
jtag_TDO_data	Output	1	JTAG Test Data Output
jtag_TDO_driven	Output	1	JTAG Test Data Output Enable
jtag_reset	Input	1	Active-high Reset
jtag_mfr_id	Input	11	The SoC Manufacturer ID which will be reported by the JTAG IDCODE instruction.

Table 5.8: SiFive standard JTAG interface for off-chip external TAPC and on-chip embedded TAPC.

Chapter 6

Memory Map

The overall physical memory map of the U54-MC Core Complex RISC-V Core IP series is shown in Tables 6.1. The U54-MC Core Complex is configured with a 38-bit physical address space.

Base	Top	Attr.	Description	Notes
0x0000_0000	0x0000_00FF		*Reserved*	Debug Address Space
0x0000_0100	0x0000_0FFF	RWX	Debug	
0x0000_1000	0x00FF_FFFF		*Reserved*	
0x0100_0000	0x0100_1FFF	RWX	E51 DTIM (8 KiB)	On Core Complex Address Space
0x0100_2000	0x017F_FFFF		*Reserved*	
0x0180_0000	0x0180_1FFF	RWX	E51 Hart 0 ITIM	
0x0180_2000	0x0180_7FFF		*Reserved*	
0x0180_8000	0x0180_EFFF	RWX	U54 Hart 1 ITIM	
0x0180_F000	0x0180_FFFF		*Reserved*	
0x0181_0000	0x0181_6FFF	RWX	U54 Hart 2 ITIM	
0x0181_7000	0x0181_7FFF		*Reserved*	
0x0181_8000	0x0181_EFFF	RWX	U54 Hart 3 ITIM	
0x0181_F000	0x0181_FFFF		*Reserved*	
0x0182_0000	0x0182_6FFF	RWX	U54 Hart 4 ITIM	
0x0182_7000	0x01FF_FFFF		*Reserved*	
0x0200_0000	0x0200_FFFF	RW	CLINT	
0x0201_0000	0x0201_0FFF	RW	Cache Controller	
0x0201_1000	0x07FF_FFFF		*Reserved*	
0x0800_0000	0x09FF_FFFF	RWX	L2-LIM	
0x0A00_0000	0x0BFF_FFFF	RWXC	L2 Zero Device	
0x0C00_0000	0x0FFF_FFFF	RW	PLIC	
0x1000_0000	0x1FFF_FFFF		*Reserved*	
0x2000_0000	0x4FFF_FFFF	RWX	Peripheral Port	Off Core Complex Address Space
0x5000_0000	0x7FFF_FFFF	RWXC	System Port	
0x8000_0000	0xFFFF_FFFF	RWXC	Memory Port	
0x01_0000_0000	0x0F_FFFF_FFFF	RWX	Peripheral Port	
0x10_0000_0000	0x1F_FFFF_FFFF	RWX	System Port	
0x20_0000_0000	0x3F_FFFF_FFFF	RWXC	Memory Port	

Table 6.1: U54-MC Core Complex Physical Memory Map.
Memory Attributes: **R** - Read **W** - Write **X** - Execute **C** - Cacheable

Chapter 7

Interrupts

This chapter describes how interrupt concepts in the RISC-V architecture apply to the U54-MC Core Complex. The definitive resource for information about the RISC-V interrupt architecture is The RISC-V Instruction Set Manual, Volume II: Privileged Architecture, Version 1.10 [2].

7.1 Interrupt Concepts

Each hart in SiFive RISC-V Core IP has support for the following interrupts: local (including software and timer), and global.

Local interrupts are signaled directly to an individual hart with a dedicated interrupt value. This allows for reduced interrupt latency as there is no arbitration required to determine which hart will service a given request, nor additional memory accesses required to determine the cause of the interrupt. Software and timer interrupts are local interrupts generated by the Core Local Interruptor (CLINT).

Global interrupts by contrast, are routed through a Platform-Level Interrupt Controller (PLIC), which can direct interrupts to any hart in the system via the external interrupt. Decoupling global interrupts from the hart(s) allows the design of the PLIC to be tailored to the platform, permitting a broad range of attributes like the number of interrupts and the prioritization and routing schemes.

By default all interrupts are are handled in machine mode. In SiFive RISC-V Core IP IP which support supervisor mode, it is possible to selectively delegate interrupts to supervisor mode.

This chapter describes the U54-MC Core Complex interrupt architecture. Chapter 8 describes the global interrupt architecture and the PLIC design. Chapter 9 describes the Core Local Interruptor.

The U54-MC Core Complex interrupt architecture is depicted in Figure 7.1.

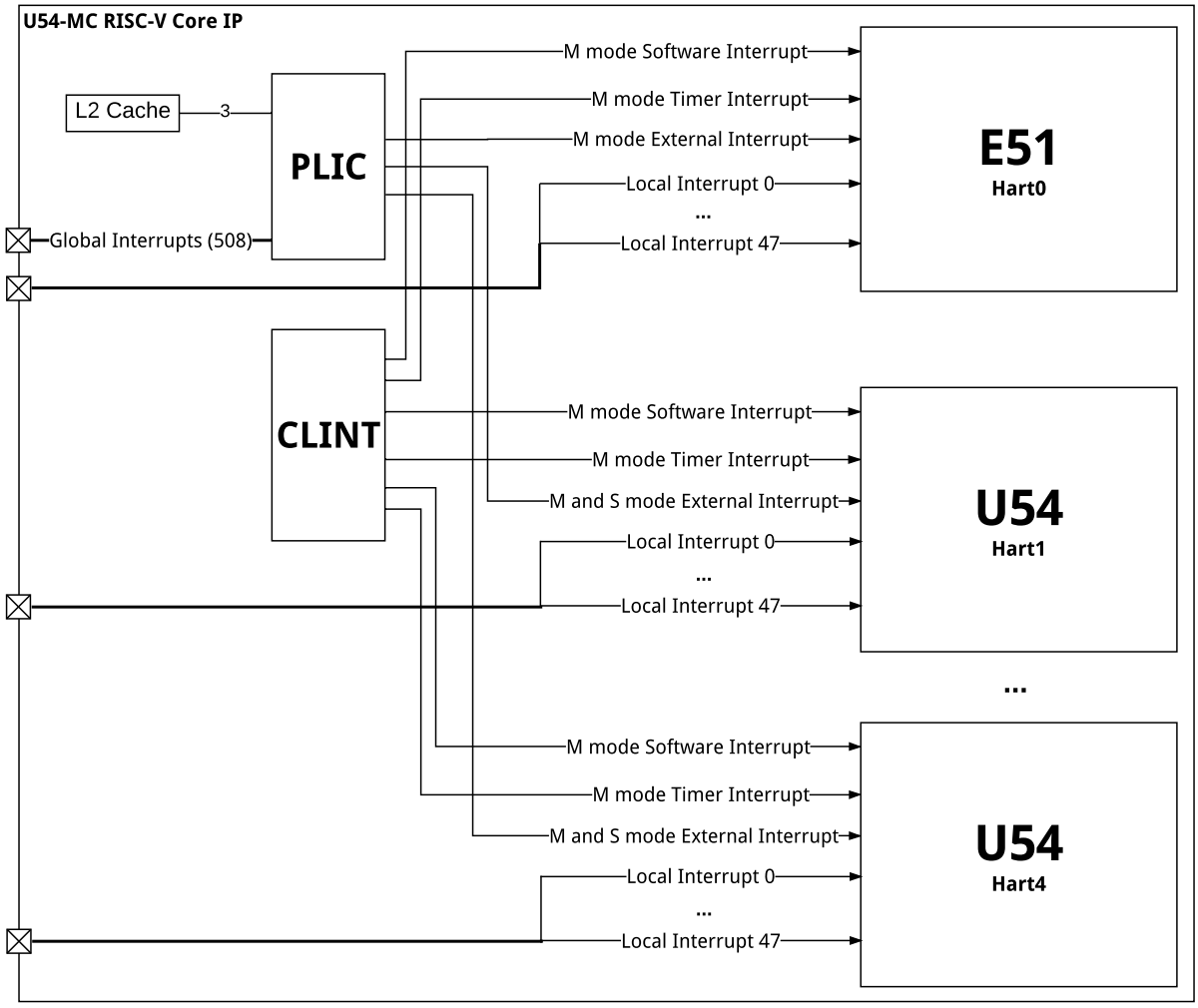

Figure 7.1: U54-MC Core Complex Interrupt Architecture Block Diagram.

7.2 Interrupt Entry and Exit

When a RISC-V hart takes an interrupt the following will occur:

- The value of `mstatus.MIE` is copied into `mstatus.MPIE`, then `mstatus.MIE` is cleared, effectively disabling interrupts.
- The current `pc` is copied into the `mepc` register, and then `pc` is set to the value of `mtvec`. In the case where vectored interrupts are enabled, `pc` is set to `mtvec.BASE` + 4×exception code.
- The privilege mode prior to the interrupt is encoded in `mstatus.MPP`.

At this point control is handed over to software in the interrupt handler with interrupts disabled. Interrupts can be re-enabled by explicitly setting `mstatus.MIE`, or by executing an MRET instruction to exit the handler. When an MRET instruction is executed, the following will occur:

- The privilege mode is set to the value encoded in `mstatus.MPP`.
- The value of `mstatus.MPIE` is copied into `mstatus.MIE`.
- The `pc` is set to the value of `mepc`.

At this point control is handed over to software.

The Control and Status Registers involved in handling RISC-V interrupts are described in Section 7.3.

7.3 Interrupt Control Status Registers

The SiFive U54-MC Core Complex specific implementation of interrupt CSRs is described below. For a complete description of RISC-V interrupt behavior and how to access CSRs, please consult The RISC-V Instruction Set Manual, Volume II: Privileged Architecture, Version 1.10 [2].

7.3.1 Machine Status Register (`mstatus`)

The `mstatus` register keeps track of and controls the hart's current operating state including whether or not interrupts are enabled. A summary of the `mstatus` fields related to interrupts in the U54-MC Core Complex is provided in Table 7.1; note that this is not a complete description of `mstatus` as it contains fields unrelated to interrupts. For the full description of `mstatus` please consult the The RISC-V Instruction Set Manual, Volume II: Privileged Architecture, Version 1.10 [2].

Machine Status Register			
CSR	`mstatus`		
Bits	**Field Name**	**Attr.**	**Description**
0	*Reserved*	WPRI	
1	SIE	RW	Supervisor Interrupt Enable
2	*Reserved*	WPRI	
3	MIE	RW	Machine Interrupt Enable
4	*Reserved*	WPRI	
5	SPIE	RW	Supervisor Previous Interrupt Enable
6	*Reserved*	WPRI	
7	MPIE	RW	Machine Previous Interrupt Enable
8	SPP	RW	Supervisor Previous Privilege Mode
[10:9]	*Reserved*	WPRI	
[12:11]	MPP	RW	Machine Previous Privilege Mode

Table 7.1: U54-MC Core Complex `mstatus` register (partial)

Interrupts are enabled by setting the MIE bit in `mstatus` and by enabling the desired individual interrupt in the `mie` register described in Section 7.3.2.

7.3.2 Machine Interrupt Enable Register (`mie`)

Individual interrupts are enabled by setting the appropriate bit in the `mie` register. The U54-MC Core Complex `mie` register is described in Table 7.2.

Machine Interrupt Enable Register			
CSR	`mie`		
Bits	**Field Name**	**Attr.**	**Description**
0	*Reserved*	WIRI	
1	SSIE	RW	Supervisor Software Interrupt Enable
2	*Reserved*	WIRI	
3	MSIE	RW	Machine Software Interrupt Enable
4	*Reserved*	WIRI	
5	STIE	RW	Supervisor Timer Interrupt Enable
6	*Reserved*	WIRI	
7	MTIE	RW	Machine Timer Interrupt Enable
8	*Reserved*	WIRI	
9	SEIE	RW	Supervisor External Interrupt Enable
10	*Reserved*	WIRI	
11	MEIE	RW	Machine External Interrupt Enable
[15:12]	*Reserved*	WPRI	
16	LIE0	RW	Local Interrupt 0 Enable
17	LIE1	RW	Local Interrupt 1 Enable
18	LIE2	RW	Local Interrupt 2 Enable
...			
63	LIE47	RW	Local Interrupt 47 Enable

Table 7.2: U54-MC Core Complex `mie` register

7.3.3 Machine Interrupt Pending (`mip`)

The machine interrupt pending (`mip`) register indicates which interrupts are currently pending. The U54-MC Core Complex `mip` register is described in Table 7.3.

Machine Interrupt Pending Register			
CSR	`mip`		
Bits	**Field Name**	**Attr.**	**Description**
0	*Reserved*	WPRI	
1	SSIP	RW	Supervisor Software Interrupt Pending
2	*Reserved*	WPRI	
3	MSIP	RO	Machine Software Interrupt Pending
4	*Reserved*	WPRI	
5	STIP	RW	Supervisor Timer Interrupt Pending
6	*Reserved*	WPRI	
7	MTIP	RO	Machine Timer Interrupt Pending
8	*Reserved*	WPRI	
9	SEIP	RW	Supervisor External Interrupt Pending
10	*Reserved*	WPRI	
11	MEIP	RO	Machine External Interrupt Pending
[15:12]	*Reserved*	WPRI	
16	LIP0	RO	Local Interrupt 0 Pending
17	LIP1	RO	Local Interrupt 1 Pending
18	LIP2	RO	Local Interrupt 2 Pending
...			
63	LIP47	RO	Local Interrupt 47 Pending

Table 7.3: U54-MC Core Complex `mip` register

7.3.4 Machine Cause Register (`mcause`)

When a trap is taken in machine mode, `mcause` is written with a code indicating the event that caused the trap. When the event that caused the trap is an interrupt, the most-significant bit of `mcause` is set to 1, and the least-significant bits indicate the interrupt number, using the same encoding as the bit positions in `mip`. For example, a Machine Timer Interrupt causes `mcause` to be set to `0x8000_0000_0000_0007`. `mcause` is also used to indicate the cause of synchronous exceptions, in which case the most-significant bit of `mcause` is set to 0. Refer to Table 7.5 for a list of synchronous exception codes.

Machine Cause Register			
CSR	`mcause`		
Bits	**Field Name**	**Attr.**	**Description**
[62:0]	Exception Code	WLRL	A code identifying the last exception.
63	Interrupt	WARL	1 if the trap was caused by an interrupt; 0 otherwise.

Table 7.4: U54-MC Core Complex `mcause` register

Interrupt Exception Codes		
Interrupt	**Exception Code**	**Description**
1	0	*Reserved*
1	1	Supervisor software interrupt
1	2	*Reserved*
1	3	Machine software interrupt
1	4	*Reserved*
1	5	Supervisor timer interrupt
1	6	*Reserved*
1	7	Machine timer interrupt
1	8	*Reserved*
1	9	Supervisor external interrupt
1	8	*Reserved*
1	11	Machine external interrupt
1	12–15	*Reserved*
1	16	Local Interrupt 0
1	17	Local Interrupt 1
1	18–62	. . .
1	63	Local Interrupt 47
0	0	Instruction address misaligned
0	1	Instruction access fault
0	2	Illegal instruction
0	3	Breakpoint
0	4	Load address misaligned
0	5	Load access fault
0	6	Store/AMO address misaligned
0	7	Store/AMO access fault
0	8	Environment call from U-mode
0	9	Environment call from S-mode
0	10	*Reserved*
0	11	Environment call from M-mode
0	12	Instruction page fault
0	13	Load page fault
0	14	*Reserved*
0	15	Store/AMO page fault
0	16–31	*Reserved*

Table 7.5: U54-MC Core Complex `mcause` Exception Codes

7.3.5 Machine Trap Vector (mtvec)

By default, all interrupts trap to a single address defined in the mtvec register. It is up to the interrupt handler to read mcause and react accordingly. RISC-V and the U54-MC Core Complex also support the ability to optionally enable interrupt vectors. When vectoring is enabled, each interrupt defined in mie will trap to its own specific interrupt handler. This allows all local interrupts to trap to exclusive handlers. With vectoring enabled, all global interrupts will trap to a single global interrupt vector.

Vectored interrupts are enabled when the MODE field of the mtvec register is set to 1.

Machine Trap Vector Register			
CSR	mtvec		
Bits	**Field Name**	**Attr.**	**Description**
[1:0]	MODE	WARL	MODE determines whether or not interrupt vectoring is enabled. The encoding for the MODE filed is described in Table 7.7
[63:2]	BASE[63:2]	WARL	Interrupt Vector Base Address. Must be aligned on a 128-byte boundary when MODE=1. Note, BASE[1:0] is not present in this register and is implicitly 0.

Table 7.6: U54-MC Core Complex mtvec register

MODE Field Encoding mtvec.MODE		
Value	**Name**	**Description**
0	Direct	All exceptions set pc to BASE
1	Vectored	Asynchronous interrupts set pc to BASE + 4×cause.
≥2	*Reserved*	

Table 7.7: Encoding of mtvec.MODE

If vectored interrupts are disabled (mtvec.MODE=0), all interrupts trap to the mtvec.BASE address. If vectored interrupts are enabled (mtvec.MODE=1), interrupts set the pc to mtvec.BASE + 4×exception code. For example, if a machine timer interrupt is taken, the pc is set to mtvec.BASE + 0x1C. Typically, the trap vector table is populated with jump instructions to transfer control to interrupt-specific trap handlers.

In vectored interrupt mode, BASE must be 128-byte aligned.

All machine external interrupts (global interrupts) are mapped to exception code of 11. Thus, when interrupt vectoring is enabled, the pc is set to address mtvec.BASE + 0x2C for any global interrupt. See Table 7.5 for the U54-MC Core Complex interrupt exception code values.

7.4 Supervisor Mode Interrupts

The U54-MC Core Complex supports the ability to selectively direct interrupts and exceptions to supervisor mode resulting in improved performance by eliminating the need for additional mode changes.

This capability is enabled by the interrupt and exception delegation CSRs; mideleg, and medeleg respectively. Supervisor interrupts and exceptions can be managed via supervisor versions of the interrupt CSRs, specifically: stvec, sip, and sie, and scause.

Machine mode software can also directly write to the `sip` register which effectively pends an interrupt to supervisor mode. This is especially useful for timer and software interrupts as it may be desired to handle these interrupts in both machine mode, and supervisor mode.

The delegation and supervisor CSRs are described in the sections below. The definitive resource for information about RISC-V supervisor interrupts is The RISC-V Instruction Set Manual, Volume II: Privileged Architecture, Version 1.10 [2].

7.4.1 Delegation Registers

By default all traps are handled in machine mode. Machine mode software can selectively delegate interrupts and exceptions to supervisor mode by setting the corresponding bits in `mideleg` and `medeleg` CSRs. The exact mapping is provided in Table 7.8 and Table 7.9 and match the `mcause` interrupt and exception codes defined in Table 7.5.

Upon taking a delegated trap, `mcause` is copied into `scause` and `mepc` is copied into `sepc` and the hart will trap to the `stvec` address in supervisor mode.

Note that local interrupts can not be delegated to supervisor mode.

Machine Interrupt Delegation Mapping		
CSR	`mideleg`	
Bits	**Attr.**	**Description**
0	WARL	*Reserved*
1	WARL	Supervisor software interrupt
[4:2]	WARL	*Reserved*
5	WARL	Supervisor timer interrupt
[8:6]	WARL	*Reserved*
9	WARL	Supervisor external interrupt
[63:8]	WARL	*Reserved*

Table 7.8: U54-MC Core Complex `mideleg` register

Machine Exception Delegation Mapping		
CSR	`medeleg`	
Bits	**Attr.**	**Description**
0	WARL	Instruction address misaligned
1	WARL	Instruction access fault
2	WARL	Illegal instruction
3	WARL	Breakpoint
4	WARL	Load address misaligned
5	WARL	Load access fault
6	WARL	Store/AMO address misaligned
7	WARL	Store/AMO access fault
8	WARL	Environment call from U-mode
9	WARL	Environment call from S-mode
[11:10]	WARL	Reserved
12	WARL	Instruction page fault
13	WARL	Load page fault
14	WARL	Reserved
15	WARL	Store/AMO page fault exception
[63:16]	WARL	*Reserved*

Table 7.9: U54-MC Core Complex `medeleg` register

7.4.2 Supervisor Status Register (`sstatus`)

Similar to machine mode, supervisor mode has a register dedicated to keeping track of the hart's current state called `sstatus`. `sstatus` is effectively a restricted view of `mstatus`, described in Section 7.3.1, in that changes made to `sstatus` are reflected in `mstatus` and vice-versa with the exception of the machine mode fields which are not visible in `sstatus`.

A summary of the `sstatus` fields related to interrupts in the U54-MC Core Complex is provided in Table 7.1; note that this is not a complete description of `sstatus` as it contains fields unrelated to interrupts. For the full description of `sstatus` please consult the The RISC-V Instruction Set Manual, Volume II: Privileged Architecture, Version 1.10 [2].

Supervisor Status Register			
CSR	`mstatus`		
Bits	**Field Name**	**Attr.**	**Description**
0	*Reserved*	WPRI	
1	SIE	RW	Supervisor Interrupt Enable
[4:2]	*Reserved*	WPRI	
5	SPIE	RW	Supervisor Previous Interrupt Enable
[7:6]	*Reserved*	WPRI	
8	SPP	RW	Supervisor Previous Privilege Mode
[12:9]	*Reserved*	WPRI	

Table 7.10: U54-MC Core Complex `sstatus` register (partial)

Interrupts are enabled by setting the SIE bit in `sstatus` and by enabling the desired individual interrupt in the `sie` register described in Section 7.4.3.

7.4.3 Supervisor Interrupt Enable Register (sie)

Supervisor interrupts are enabled by setting the appropriate bit in the sie register. The U54-MC Core Complex sie register is described in Table 7.11.

Supervisor Interrupt Enable Register			
CSR	sie		
Bits	**Field Name**	**Attr.**	**Description**
0	*Reserved*	WIRI	
1	SSIE	RW	Supervisor Software Interrupt Enable
[4:2]	*Reserved*	WIRI	
5	STIE	RW	Supervisor Timer Interrupt Enable
[8:6]	*Reserved*	WIRI	
9	SEIE	RW	Supervisor External Interrupt Enable
[63:10]	*Reserved*	WIRI	

Table 7.11: U54-MC Core Complex sie register

7.4.4 Supervisor Interrupt Pending (sip)

The supervisor interrupt pending (sip) register indicates which interrupts are currently pending. The U54-MC Core Complex sip register is described in Table 7.12.

Supervisor Interrupt Pending Register			
CSR	mip		
Bits	**Field Name**	**Attr.**	**Description**
0	*Reserved*	WPRI	
1	SSIP	RW	Supervisor Software Interrupt Pending
[4:2]	*Reserved*	WPRI	
5	STIP	RW	Supervisor Timer Interrupt Pending
[8:6]	*Reserved*	WPRI	
9	SEIP	RW	Supervisor External Interrupt Pending
[63:10]	*Reserved*	WPRI	

Table 7.12: U54-MC Core Complex sip register

7.4.5 Supervisor Cause Register (scause)

When a trap is taken in supervisor mode, scause is written with a code indicating the event that caused the trap. When the event that caused the trap is an interrupt, the most-significant bit of scause is set to 1, and the least-significant bits indicate the interrupt number, using the same encoding as the bit positions in sip. For example, a Supervisor Timer Interrupt causes scause to be set to 0x8000_0000_0000_0005. scause is also used to indicate the cause of synchronous exceptions, in which case the most-significant bit of scause is set to 0. Refer to Table 7.14 for a list of synchronous exception codes.

Supervisor Cause Register			
CSR	`scause`		
Bits	**Field Name**	**Attr.**	**Description**
[62:0]	Exception Code	WLRL	A code identifying the last exception.
63	Interrupt	WARL	1 if the trap was caused by an interrupt; 0 otherwise.

Table 7.13: U54-MC Core Complex `scause` register

Supervisor Interrupt Exception Codes		
Interrupt	**Exception Code**	**Description**
1	0	*Reserved*
1	1	Supervisor software interrupt
1	2-4	*Reserved*
1	5	Supervisor timer interrupt
1	6-8	*Reserved*
1	9	Supervisor external interrupt
1	≥ 10	*Reserved*
0	0	Instruction address misaligned
0	1	Instruction access fault
0	2	Illegal instruction
0	3	Breakpoint
0	4	*Reserved*
0	5	Load access fault
0	6	Store/AMO address misaligned
0	7	Store/AMO access fault
0	8	Environment call from U-mode
0	9-11	*Reserved*
0	12	Instruction page fault
0	13	Load page fault
0	14	Reserved
0	15	Store/AMO page fault
0	≥ 16	Reserved

Table 7.14: U54-MC Core Complex `scause` Synchronous Exception Codes

7.4.6 Supervisor Trap Vector (stvec)

By default, all interrupts trap to a single address defined in the stvec register. It is up to the interrupt handler to read scause and react accordingly. RISC-V and the U54-MC Core Complex also support the ability to optionally enable interrupt vectors. When vectoring is enabled, each interrupt defined in sie will trap to its own specific interrupt handler.

Vectored interrupts are enabled when the MODE field of the stvec register is set to 1.

Supervisor Trap Vector Register			
CSR	stvec		
Bits	**Field Name**	**Attr.**	**Description**
[1:0]	MODE	WARL	MODE determines whether or not interrupt vectoring is enabled. The encoding for the MODE filed is described in Table 7.7
[63:2]	BASE[63:2]	WARL	Interrupt Vector Base Address. Must be aligned on a 128-byte boundary when MODE=1. Note, BASE[1:0] is not present in this register and is implicitly 0.

Table 7.15: U54-MC Core Complex stvec register

MODE Field Encoding stvec.MODE		
Value	**Name**	**Description**
0	Direct	All exceptions set pc to BASE
1	Vectored	Asynchronous interrupts set pc to BASE + 4×cause.
≥2	*Reserved*	

Table 7.16: Encoding of stvec.MODE

If vectored interrupts are disabled (stvec.MODE=0), all interrupts trap to the stvec.BASE address. If vectored interrupts are enabled (stvec.MODE=1), interrupts set the pc to stvec.BASE + 4×exception code. For example, if a supervisor timer interrupt is taken, the pc is set to stvec.BASE + 0x14. Typically, the trap vector table is populated with jump instructions to transfer control to interrupt-specific trap handlers.

In vectored interrupt mode, BASE must be 128-byte aligned.

All supervisor external interrupts (global interrupts) are mapped to exception code of 9. Thus, when interrupt vectoring is enabled, the pc is set to address stvec.BASE + 0x24 for any global interrupt.

Please see Table 7.14 for the U54-MC Core Complex supervisor mode interrupt exception code values.

7.5 Interrupt Priorities

Local interrupts have higher priority than global interrupts. As such, if a local and a global interrupt arrive at a hart on the same cycle, the local interrupt will be taken if it is enabled.

Priorities of local interrupts are determined by the local interrupt ID, with Local Interrupt 47 being highest priority. For example, if both Local Interrupt 47 and Local Interrupt 6 arrive in the same cycle, Local Interrupt 47 will be taken.

Local Interrupt 47 is the highest-priority interrupt in the U54-MC Core Complex for a given hart. Given that Local Interrupt 47's exception code is also the greatest, it occupies the last slot in the interrupt vector table. This unique position in the vector table allows for Local Interrupt 47's trap handler to be placed in-line, without the need for a jump instruction as with other interrupts when operating in vectored mode. Hence, Local Interrupt 47 should be used for the most latency-sensitive interrupt in the system for a given hart. Individual priorities of global interrupts are determined by the PLIC, as discussed in Chapter 8.

U54-MC Core Complex interrupts are prioritized as follows, in decreasing order of priority:

- Local Interrupt 47
- ...
- Local Interrupt 0
- Machine external interrupts
- Machine software interrupts
- Machine timer interrupts
- Supervisor external interrupts
- Supervisor software interrupts
- Supervisor timer interrupts

7.6 Interrupt Latency

Interrupt latency for the U54-MC Core Complex, as counted by the numbers of cycles it takes from signaling of the interrupt to the hart to the first instruction fetch of the handler, is 4 cycles.

Global interrupts routed through the PLIC incur additional latency of 3 cycles where the PLIC is clocked by `clock`. This means that the total latency, in cycles, for a global interrupt is:
$4 + 3\times$ (`core_clock` Hz$\div$`clock` Hz).

This is a best case cycle count and assumes the handler is cached or located in ITIM. It does not take into account additional latency from a peripheral source.

Additionally, the hart will not abandon a Divide instruction in flight. This means if an interrupt handler tries to use a register which is the destination register of a divide instruction, the pipeline will stall until the divide is complete.

Chapter 8

Platform-Level Interrupt Controller (PLIC)

This chapter describes the operation of the platform-level interrupt controller (PLIC) on the SiFive U54-MC Core Complex. The PLIC complies with The RISC-V Instruction Set Manual, Volume II: Privileged Architecture, Version 1.10 [2], and can support a maximum of 511 external interrupt sources with 7 priority levels.

The U54-MC Core Complex PLIC resides is in the `clock` timing domain allowing for relaxed timing requirements. The latency of global interrupts, as perceived by a hart, increases with the ratio of the `core_clock` frequency and the `clock` frequency.

8.1 Memory Map

The memory map for the SiFive U54-MC Core Complex PLIC control registers is shown in Table 8.2. The PLIC memory map has been designed to only require naturally aligned 32-bit memory accesses.

PLIC Memory Map				
Address	**Width**	**Attr.**	**Description**	**Notes**
0x0C00_0000			*Reserved*	See Section 8.3 for more information
0x0C00_0004	4B	RW	source 1 priority	
0x0C00_0008	4B	RW	source 2 priority	
. . .				
0x0C00_0800	4B	RW	source 511 priority	
0x0C00_0804			*Reserved*	
. . .				
0x0C00_0FFF				
0x0C00_1000	4B	RO	Start of pending array	See Section 8.4 for more information
. . .				
0x0C00_1014	4B	RO	Last word of pending array	
0x0C00_1018			*Reserved*	
. . .				
0x0C00_1FFF				
0x0C00_2000	4B	RW	Start of Hart 0 M-mode enables	See Section 8.5 for more information
. . .				
0x0C00_2014	4B	RW	End of Hart 0 M-mode enables	
0x0C00_2018			*Reserved*	
. . .				
0x0C00_207F				
0x0C00_2080	4B	RW	Hart 1 M-mode enables	Same layout as Hart 0 M-mode enables
. . .				
0x0C00_2094	4B	RW	End of Hart 1 M-mode enables	
0x0C00_2100	4B	RW	Hart 1 S-mode enables	
. . .				
0x0C00_2114	4B	RW	End of Hart 1 S-mode enables	
0x0C00_2180	4B	RW	Hart 2 M-mode enables	
. . .				
0x0C00_2194	4B	RW	End of Hart 2 M-mode enables	
0x0C00_2200	4B	RW	Hart 2 S-mode enables	
. . .				
0x0C00_2214	4B	RW	End of Hart 2 S-mode enables	
0x0C00_2280	4B	RW	Hart 3 M-mode enables	
. . .				
0x0C00_2294	4B	RW	End of Hart 3 M-mode enables	
0x0C00_2300	4B	RW	Hart 3 S-mode enables	
. . .				
0x0C00_2314	4B	RW	End of Hart 3 S-mode enables	
0x0C00_2380	4B	RW	Hart 4 M-mode enables	
. . .				
0x0C00_2394	4B	RW	End of Hart 4 M-mode enables	
0x0C00_2400	4B	RW	Hart 4 S-mode enables	
. . .				
0x0C00_2414	4B	RW	End of Hart 4 S-mode enables	
Continued on next page.				

Table 8.1: SiFive PLIC Register Map. Only naturally aligned 32-bit memory accesses are supported.

PLIC Register Map (Continued)				
Address	**Width**	**Attr.**	**Description**	**Notes**
0x0C00_2480 ... 0x0C1F_FFFF			*Reserved*	
0x0C20_0000	4B	RW	Hart 0 M-mode priority threshold	See Section 8.6 and Section 8.7 for more information
0x0C20_0004	4B	RW	Hart 0 M-mode claim/complete	
0x0C20_1000	4B	RW	Hart 1 M-mode priority threshold	
0x0C20_1004	4B	RW	Hart 1 M-mode claim/complete	
0x0C20_2000	4B	RW	Hart 1 S-mode priority threshold	
0x0C20_2004	4B	RW	Hart 1 S-mode claim/complete	
0x0C20_3000	4B	RW	Hart 2 M-mode priority threshold	
0x0C20_3004	4B	RW	Hart 2 M-mode claim/complete	
0x0C20_4000	4B	RW	Hart 2 S-mode priority threshold	
0x0C20_4004	4B	RW	Hart 2 S-mode claim/complete	
0x0C20_5000	4B	RW	Hart 3 M-mode priority threshold	
0x0C20_5004	4B	RW	Hart 3 M-mode claim/complete	
0x0C20_6000	4B	RW	Hart 3 S-mode priority threshold	
0x0C20_6004	4B	RW	Hart 3 S-mode claim/complete	
0x0C20_7000	4B	RW	Hart 4 M-mode priority threshold	
0x0C20_7004	4B	RW	Hart 4 M-mode claim/complete	
0x0C20_8000	4B	RW	Hart 4 S-mode priority threshold	
0x0C20_8004	4B	RW	Hart 4 S-mode claim/complete	

Table 8.2: SiFive PLIC Register Map Continued. Only naturally aligned 32-bit memory accesses are supported.

8.2 Interrupt Sources

The U54-MC Core Complex is delivered with several pre-integrated, on core complex peripherals which have interrupt signals already connected to the PLIC. The mapping of these on core complex peripheral interrupts to their corresponding ID's are provided in Table 8.3.

PLIC Interrupt ID Mapping		
IRQ	**Peripheral**	**Description**
0	N/A	Global Interrupt 0 is defined to mean "no interrupt" and is not connected to a peripheral.
1	Off Core Complex	Connected to `global_interrupt[0]` signal described in Section 5.4.
2	Off Core Complex	Connected to `global_interrupt[1]` signal described in Section 5.4.
...		
508	Off Core Complex	Connected to `global_interrupt[507]` signal described in Section 5.4.
509	L2 Cache Controller	Connected to `DirError` and interrupts when a L2 meta-data error has occurred.
510	L2 Cache Controller	Connected to `DataError` and interrupts when a L2 data error has occurred.
511	L2 Cache Controller	Connected to `DataFail` and interrupts when an uncorrectable L2 data error has occurred.

Table 8.3: PLIC Interrupt ID Mapping

The U54-MC Core Complex has 511 interrupt sources exposed at the top level via the `global_interrupts` signals. These signals are positive-level triggered.

Any unused `global_interrupts` inputs should be tied to logic 0.

In the PLIC, as specified in The RISC-V Instruction Set Manual, Volume II: Privileged Architecture, Version 1.10 [2], Global Interrupt ID 0 is defined to mean "no interrupt" , hence `global_interrupts[0]` corresponds to PLIC Interrupt ID 1.

8.3 Interrupt Priorities

Each PLIC interrupt source can be assigned a priority by writing to its 32-bit memory-mapped `priority` register. The U54-MC Core Complex supports 7 levels of priority. A priority value of 0 is reserved to mean "never interrupt" and effectively dissables the interrupt. Priority 1 is the lowest active priority, and priority 7 is the highest. Ties between global interrupts of the same priority are broken by the Interrupt ID; interrupts with the lowest ID have the highest effective priority. Please see Table 8.4 for the detailed register description.

PLIC Interrupt Priority Register (priority)				
Base Address		0x0C00_0000 + 4×Interrupt ID		
Bits	**Field Name**	**Attr.**	**Rst.**	**Description**
[2:0]	Priority	WARL	X	Sets the priority for a given global interrupt.
31:3]	*Reserved*	WIRI	X	

Table 8.4: PLIC Interrupt Priority Registers

8.4 Interrupt Pending Bits

The current status of the interrupt source pending bits in the PLIC core can be read from the pending array, organized as 15 words of 32 bits. The pending bit for interrupt ID N is stored in bit $(N \text{ mod } 32)$ of word $(N/32)$. As such, the U54-MC Core Complex has 15 interrupt pending registers. Bit 0 of word 0, which represents the non-existent interrupt source 0, is hardwired to zero.

A pending bit in the PLIC core can be cleared by setting the associated enable bit then performing a claim as as described in Section 8.7.

PLIC Interrupt Pending Register 1 (pending1)				
Base Address		0x0C00_1000		
Bits	**Field Name**	**Attr.**	**Rst.**	**Description**
0	Interrupt 0 Pending	RO	0	Non-existent global interrupt 0 is hardwired to zero
1	Interrupt 1 Pending	RO	0	Pending bit for global interrupt 1
2	Interrupt 2 Pending	RO	0	Pending bit for global interrupt 2
...				
31	Interrupt 31 Pending	RO	0	Pending bit for global interrupt 31

Table 8.5: PLIC Interrupt Pending Register 1

PLIC Interrupt Pending Register 15 (pending15)				
Base Address		0x0C00_103C		
Bits	**Field Name**	**Attr.**	**Rst.**	**Description**
1	Interrupt 480 Pending	RO	0	Pending bit for global interrupt 480
...				
31	Interrupt 511 Pending	RO	0	Pending bit for global interrupt plicinputs

Table 8.6: PLIC Interrupt Pending Register 15

8.5 Interrupt Enables

Each global interrupt can be enabled by setting the corresponding bit in the enables register. The enables registers are accessed as a contiguous array of 15×32-bit words, packed the same way as the pending bits. Bit 0 of enable word 0 represents the non-existent interrupt ID 0 and is hardwired to 0.

64-bit and 32-bit word accesses are supported by the enables array in SiFive RV64 systems.

PLIC Interrupt Enable Register 1 (enable1)				
Base Address		0x0C00_2000		
Bits	**Field Name**	**Attr.**	**Rst.**	**Description**
0	Interrupt 0 Enable	RW	X	Non-existent global interrupt 0 is hardwired to zero
1	Interrupt 1 Enaable	RW	X	Enable bit for global interrupt 1
2	Interrupt 2 Enaable	RW	X	Enable bit for global interrupt 2
...				
31	Interrupt 31 Enaable	RW	X	Enable bit for global interrupt 31

Table 8.7: PLIC Interrupt Enable Register 1

PLIC Interrupt Enable Register 15 (enable15)				
Base Address		0x0C00_203C		
Bits	**Field Name**	**Attr.**	**Rst.**	**Description**
0	Interrupt 480 Enable	RW	X	Enable bit for global interrupt 480
...				
31	Interrupt 511 Enable	RW	X	Enable bit for global interrupt 511

Table 8.8: PLIC Interrupt Enable Register 15

8.6 Priority Thresholds

The U54-MC Core Complex supports setting of a interrupt priority threshold via the `threshold` register. The `threshold` is a **WARL** field, where the U54-MC Core Complex supports a maximum threshold of 7.

The U54-MC Core Complex will mask all PLIC interrupts of a priority less than or equal to `threshold`. For example, a `threshold` value of zero permits all interrupts with non-zero priority, whereas a value of 7 masks all interrupts.

PLIC Interrupt Priority Threshold Register (threshold)				
Base Address		0x0C20_0000		
Bits	**Field Name**	**Attr.**	**Rst.**	**Description**
[2:0]	Threshold	RW	X	Sets the priority threshold
[31:3]	*Reserved*	WIRI	X	

Table 8.9: PLIC Interrupt Threshold Registers

8.7 Interrupt Claim Process

The U54-MC Core Complex can perform an interrupt claim by reading the `claim/complete` register (Table 8.10), which returns the ID of the highest priority pending interrupt or zero if there is no pending interrupt. A successful claim will also atomically clear the corresponding pending bit on the interrupt source.

The U54-MC Core Complex can perform a claim at any time, even if the MEIP bit in the `mip` (Section 7.3.3) register is not set.

The claim operation is not affected by the setting of the priority threshold register.

8.8 Interrupt Completion

The U54-MC Core Complex signals it has completed executing an interrupt handler by writing the interrupt ID it received from the claim to the `claim/complete` register (Table 8.10). The PLIC does not check whether the completion ID is the same as the last claim ID for that target. If the completion ID does not match an interrupt source that is currently enabled for the target, the completion is silently ignored.

PLIC Claim/Complete Register (`claim`)				
Base Address		`0x0C20_0004`		
Bits	**Field Name**	**Attr.**	**Rst.**	**Description**
[31:0]	Interrupt Claim	RW	X	A read of zero indicates that no interrupts are pending. A non-zero read contains the id of the highest pending interrupt. A write to this register signals completion of the interrupt id written

Table 8.10: PLIC Interrupt Claim/Complete Register

Chapter 9

Core Local Interruptor (CLINT)

The CLINT block holds memory-mapped control and status registers associated with software and timer interrupts. The U54-MC Core Complex CLINT complies with The RISC-V Instruction Set Manual, Volume II: Privileged Architecture, Version 1.10 [2].

9.1 U54-MC Core Complex CLINT Address Map

Table 9.1 shows the memory map for CLINT on SiFive U54-MC Core Complex.

CLINT Register Map				
Address	**Width**	**Attr.**	**Description**	**Notes**
0x0200_0000	4B	RW	`msip` for hart 0	MSIP Registers
0x0200_0004	4B	RW	`msip` for hart 1	
0x0200_0008	4B	RW	`msip` for hart 2	
0x0200_000C	4B	RW	`msip` for hart 3	
0x0200_0010	4B	RW	`msip` for hart 4	
0x0200_0014 ... 0x0200_3FFF			*Reserved*	
0x0200_4000	8B	RW	`mtimecmp` for hart 0	Timer compare register
0x0200_4008	8B	RW	`mtimecmp` for hart 1	
0x0200_4010	8B	RW	`mtimecmp` for hart 2	
0x0200_4018	8B	RW	`mtimecmp` for hart 3	
0x0200_4020	8B	RW	`mtimecmp` for hart 4	
0x0200_4028 ... 0x0200_BFF7			*Reserved*	
0x0200_BFF8	8B	RO	`mtime`	Timer register
0x0200_C000 ... 0x0200_FFFF			*Reserved*	

Table 9.1: SiFive U54-MC Core Complex CLINT Memory Map.

9.2 MSIP Registers

Machine-mode software interrupts are generated by writing to the memory-mapped control register `msip`. The `msip` register is a 32-bit wide **WARL** register, where the LSB is reflected in the `msip` bit of the `mip` register. Other bits in the `msip` registers are hardwired to zero. On reset, the `msip` registers are cleared to zero.

Software interrupts are most useful for interprocessor communication in multi-hart systems, as harts may write each other's `msip` bits to effect interprocessor interrupts.

9.3 Timer Registers

`mtime` is a 64-bit read-write register that contains the number of cycles counted from the `rtc_toggle` signal described in Chapter 5. A timer interrupt is pending whenever `mtime` is greater than or equal to the value in the `mtimecmp` register. The timer interrupt is reflected in the `mtip` bit of the `mip` register described in Chapter 7.

On reset, `mtime` is cleared to zero. The `mtimecmp` registers are not reset.

9.4 Supervisor Mode Delegation

By default all interrupts trap to machine mode including timer and software interrupts. In order for supervisor timer and software interrupts to trap directly to supervisor mode, supervisor timer and software interrupts must first be delegated to supervisor mode.

Please see Chapter 7 Section 7.4 for more details on supervisor mode interrupts.

Chapter 10

Physical Memory Protection

This chapter describes how physical memory protection concepts in the RISC-V architecture apply to the U54-MC Core Complex. The definitive resource for information about the RISC-V physical memory protection is The RISC-V Instruction Set Manual, Volume II: Privileged Architecture, Version 1.10 [2].

10.1 Functional Description

The U54-MC Core Complex includes a Physical Memory Protection (PMP) unit, which can be used to restrict access to memory and isolate processes from each other.

The U54-MC Core Complex PMP unit has 8 regions and a minimum granularity of 4 bytes. It is permitted to have overlapping regions. The U54-MC Core Complex PMP unit implements the architecturally defined `pmpcfg0` CSR, supporting 8 regions. `pmpcg1`, `pmpcfg2` and `pmpcfg3` are implemented but hardwired to zero.

The PMP registers may only be programmed in M-mode. Ordinarily, the PMP unit enforces permissions on S-mode and U-mode accesses. However, locked regions (see Section 10.2) additionally enforce their permissions on M-mode.

10.2 Region Locking

The PMP allows for region locking whereby once a region is locked, further writes to the configuration and address registers are ignored. Locked PMP entries may only be unlocked with a system reset. A region may be locked by setting the `L` bit in the pmp`i`cfg register.

In addition to locking the PMP entry, the `L` bit indicates whether the R/W/X permissions are enforced on M-Mode accesses. When the `L` bit is set, these permissions are enforced for all privilege modes. When `L` bit is clear, the R/W/X permissions apply only to U-mode.

When implementing less than he maximum DTIM RAM, it is necessary to lock one PMP region encompassing the unimplemented address space with no R/W/X permissions. Doing so will force all access to the unimplemented address space to generate an exception.

For example, if one only implemented 32 KiB of DTIM RAM, then setting `pmp0cfg`=0x98 and `pmpaddr0`=0x2000_0FFF will disable access to the unimplemented 32 KiB region above.

Chapter 11

Level 2 Cache Controller

This chapter will describe the functionality of the Level 2 Cache Controller used in the U54-MC Core Complex.

11.1 Level 2 Cache Controller Overview

The SiFive Level 2 Cache Controller is used to provide access to fast copies of memory for masters in a Core Complex. The Level 2 Cache Controller also acts as directory-based coherency manager.

The SiFive Level 2 Cache Controller offers extensive flexibility as it allows for programmatically enabling cache ways, way masking per master, ECC support with error tracking statistics, error injection, and interrupt signaling capabilities.

All of these features are described in Section 11.2.

11.2 Functional Description

The U54-MC Core Complex L2 Cache Controller is configured into 4 banks where each bank contains 512 sets of 16 ways and each way containing a 64 Byte block. This subdivision into banks helps facilitate increased available bandwidth between CPU masters and the L2 Cache as each bank has its own 128-bit TL-C inner port. As such, multiple requests to different banks may proceed in parallel.

The outer port of the L2 Cache Controller is a 128-bit TL-C port shared amongst all banks and typically connected to a DDR controller. This port is described in Chapter 5.

11.2.1 Error Correcting Codes (ECC)

The SiFive Level 2 Cache Controller supports ECC allowing for correction of single bit errors, and detection of double bit errors (SECDEC) on L2 data. The Cache Controller also has ECC for meta-data information (index and tag information) where it can correct a single bit error.

Whenever a correctable error is detected, the caches immediately repair the corrupted bit and write it back to SRAM. This corrective procedure is completely invisible to application software. However, to support diagnostics, the cache records the address of the most recently corrected meta-data and data errors. Whenever a new error is corrected, a counter is increased and an

interrupt is raised. There are independent addresses, counters, and interrupts for correctable meta-data and data errors.

`DirError`, `DataError`, `DataFail` signals are used to indicate that an L2 meta-data, data, or uncorrectable L2 data error error has occurred respectively. These signals are connected to the PLIC as described in Chapter 8 and are cleared upon reading their respective count registers.

11.2.2 Way Enable and Masking

Similar to the ITIM discussed in Chapter 3, the SiFive Level 2 Cache Controller allows for its SRAMs to act either as direct addressed memory in the RISC-V Core IP address space, or as a cache which is controlled by the L2 Cache Controller and can contain a copy of any cacheable address.

When cache ways are disabled, they are addressable in the L2-LIM (L2 Loosely Integrated Memory) address space as described in the U54-MC Core Complex memory map in Chapter 6. Fetching instructions or data from the L2-LIM provides deterministic behavior equivalent to an L2 cache hit, with no possibility of a cache miss. Accesses to L2-LIM are always given priority over cache way accesses which target the same L2 cache bank.

Out of reset all ways, except for way 0, are disabled. Cache ways can be enabled by writing to the `WayEnable` register described in Section 11.4.2. Once a cache way is enabled, it can not be disabled unless the RISC-V Core IP is reset.

Additionally, it is possible to restrict the amount of cache memory a CPU master is able use by using the `WayMaskX` register described in Section 11.4.11.

11.3 Memory Map

The L2 Cache Controller memory map is shown in Table 11.1.

Level 2 Cache Controller Memory Map				
Offset	**Width**	**Attr.**	**Description**	**Notes**
0x000	4B	RO	`Config`	Information on the configuration of the L2 Cache
0x008	1B	RW	`WayEnable`	Way enable register
0x040	4B	WO	`ECCInjectError`	ECC error injection Register
0x100	8B	RO	`ECCDirFixAddr`	Address of most recently corrected meta-data error
0x108	4B	RO	`ECCDirFixCount`	Count of corrected meta-data errors
0x140	8B	RO	`ECCDataFixAddr`	Address of most recently corrected data error
0x148	4B	RO	`ECCDataFixCount`	Count of corrected data errors
0x160	8B	RO	`ECCDataFailAddr`	Address of most recent uncorrectable data error
0x168	4B	RO	`ECCDataFailCount`	Count of uncorrectable data errors
0x200	8B	WO	`Flush64`	Flush cache block, 64-bit address
0x240	4B	WO	`Flush32`	Flush cache block, 32-bit address
0x800	8B	RW	`WayMask0`	Master 0 way mask register
0x808	8B	RW	`WayMask1`	Master 1 way mask register
0x810	8B	RW	`WayMask2`	Master 2 way mask register
0x818	8B	RW	`WayMask3`	Master 3 way mask register
0x820	8B	RW	`WayMask4`	Master 4 way mask register

Table 11.1: Level 2 Cache Controller Memory Map

11.4 Register Descriptions

This section will describe the functionality of the memory mapped registers in the Level 2 Cache Controller

11.4.1 Cache Configuration Register `Config`

The `Config` Register can be used to programmatically determine information regarding the cache size and organization.

Cache Configuration Register (Config)				
Register Offset		0x000		
Bits	**Field Name**	**Attr.**	**Rst.**	**Description**
[7:0]	Banks	RO	4	Returns the number of banks in the cache
[15:8]	Ways	RO	16	Returns the number of ways in the cache
[23:16]	Sets	RO	9	Returns the Base-2 logarithm of the number of sets in a cache bank
[31:24]	Bytes	RO	6	Returns the Base-2 logarithm of the number of bytes in cache blocks

Table 11.2: Cache Configuration Register

11.4.2 Way Enable Register WayEnable

The WayEnable register determines which ways of the Level 2 Cache Controller are enabled as Cache. Cache ways which are not enabled, are mapped into the U54-MC Core Complex's L2-LIM (Loosely Integrated Memory) as described in the memory map in Chapter 6.

This register is initialized to 0 on reset and may only be increased. This means that, out of reset, only a single L2 cache way is enabled as one cache way must always remain enabled. Once a cache way is enabled, the only way to map it back into the L2-LIM address space is by a reset.

Way Enable Register (WayEnable)				
Register Offset		0x008		
Bits	**Field Name**	**Attr.**	**Rst.**	**Description**
[7:0]	Way Enable	RW	0	Way indexes less than or equal to this register value may be used by the cache
[63:8]	*Reserved*	RW		

Table 11.3: Way Enable Register

11.4.3 ECC Error Injection Register ECCInjectError

The ECCInjectError register can be used to insert a an ECC error into either the backing data or meta-data SRAM. This function can be used to test error correction logic, measurement, and recovery.

ECC Error Injection Register (ECCInjectError)				
Register Offset		0x040		
Bits	**Field Name**	**Attr.**	**Rst.**	**Description**
[7:0]	Bit Position	RW	0	Specifies a bit position to toggle, within an SRAM. The SRAM width depends on the microarchitecture, but is typically 72 bits for data SRAMs and ≈ 24 bits for Directory SRAM
[15:8]	*Reserved*	RW		
16	Target	RW	0	Setting this bit means the error injection will target the meta-data SRAMs, otherwise the error injection will target the data SRAMs
[31:17]	*Reserved*	RW		

Table 11.4: ECC Error Injection Register

11.4.4 ECC Directory Fix Address ECCDirFixAddr

The ECCDirFixAddr register is a Read Only register which contains the address of the most recently corrected meta-data error. This field only supplys the portions of the address which corrspond to the affected set and bank, since all ways are corrected together.

11.4.5 ECC Directory Fix Count ECCDirFixCount

The ECCDirFixCount register is a Read Only register which contains the number of corrected L2 meta-data errors.

Reading this register clears the DirError interrupt signal described in Section 11.2.1.

11.4.6 ECC Data Fix Address ECCDataFixAddr

The ECCDataFixAddr register is a Read Only register which contains the address of the most recently corrected L2 data error.

11.4.7 ECC Data Fix Count ECCDataFixCount

The ECCDataFixCount register is a Read Only register which contains the number of corrected data errors.

Reading this register clears the DataError interrupt signal described in Section 11.2.1.

11.4.8 ECC Data Fail Address ECCDataFailAddr

The ECCDataFailAddr register is a Read Only register which contains the address of the most recent un-corrected L2 data error.

11.4.9 ECC Data Fail Count ECCDataFailCount

The ECCDataFailCount register is a Read Only register which contains the number of un-corrected data errors.

Reading this register clears the DataFail interrupt signal described in Section 11.2.1.

11.4.10 Cache Flush Registers

The U54-MC Core Complex L2 Cache Controller provides two registers which can be used for flushing specific cache blocks.

`Flush64` is a 64-bit write only register that will flush the cache block containing the address written. `Flush32` is a 32-bit write only register that will flush a cache block containing the written address left shifted by 4 bytes. In both registers, all bits must be written in a single access for the flush to take effect.

11.4.11 Way Mask Registers `WayMaskX`

The `WayMaskX` register allows a master connected to the L2 Cache Controller to specify which L2 cache ways can be evicted by master 'X' as specified in the `WayMaskX` register. All masters can still use data previously stored in any of the ways. The mask serves to limit the amount of cache memory a given master can monopolize for its own use.

At least one cache way must be enabled. It is recommended to set/clear bits in this register using atomic operations.

Way Mask Register (`WayMaskX`)				
Register Offset		`0x800+`$(8 \times MasterID)$		
Bits	**Field Name**	**Attr.**	**Rst.**	**Description**
0	Way0 Enable	RW	1	Setting this bit enables L2 Cache Way 0
1	Way1 Enable	RW	1	Setting this bit enables L2 Cache Way 1
...				
15	Way15 Enable	RW	1	Setting this bit enables L2 Cache Way 15
[63:16]	*Reserved*	RW	1	

Table 11.5: Way Mask Register

Chapter 12

Debug

This chapter describes the operation of SiFive debug hardware, which follows the RISC-V Debug Specification v0p13. Currently only interactive debug and hardware breakpoints are supported.

12.1 Debug CSRs

This section describes the per-hart trace and debug registers (TDRs), which are mapped into the CSR space as follows:

CSR Name	Description	Allowed Access Modes
tselect	Trace and debug register select	D, M
tdata1	First field of selected TDR	D, M
tdata2	Second field of selected TDR	D, M
tdata3	Third field of selected TDR	D, M
dcsr	Debug control and status register	D
dpc	Debug PC	D
dscratch	Debug scratch register	D

The dcsr, dpc, and dscratch registers are only accessible in debug mode, while the tselect and tdata1–3 registers are accessible from either debug mode or machine mode.

12.1.1 Trace and Debug Register Select (tselect)

To support a large and variable number of TDRs for tracing and breakpoints, they are accessed through one level of indirection where the tselect register selects which bank of three tdata1–3 registers are accessed via the other three addresses.

The tselect register has the format shown below:

The index field is a **WARL** field that will not hold indices of unimplemented TDRs. Even if index can hold a TDR index, it does not guarantee the TDR exists. The type field of tdata1 must be inspected to determine whether the TDR exists.

Trace and Debug Select Register			
CSR	tselect		
Bits	**Field Name**	**Attr.**	**Description**
[31:0]	index	WARL	Selection index of trace and debug registers

Table 12.1: U54-MC Core Complex tselect CSR.

12.1.2 Test and Debug Data Registers (tdata1–3)

The tdata1–3 registers are XLEN-bit read/write registers selected from a larger underlying bank of TDR registers by the tselect register.

Trace and Debug Data Register 1			
CSR	tdata1		
Bits	**Field Name**	**Attr.**	**Description**
[27:0]	TDR-Specific Data		
[31:28]	type	RO	Type of the trace & debug register selected by tselect

Table 12.2: U54-MC Core Complex tdata1 CSR.

Trace and Debug Data Registers 2 and 3			
CSR	tdata2/3		
Bits	**Field Name**	**Attr.**	**Description**
[31:0]	type		TDR-Specific Data

Table 12.3: U54-MC Core Complex tdata2/3 CSRs.

The high nibble of tdata1 contains a 4-bit type code that is used to identify the type of TDR selected by tselect. The currently defined types are shown below:

type	Description
0	No such TDR register
1	Reserved
2	Address/Data Match Trigger
$\geq$3	Reserved

The dmode bit selects between debug mode (dmode=1) and machine mode (dmode=1) views of the registers, where only debug mode code can access the debug mode view of the TDRs. Any attempt to read/write the tdata1–3 registers in machine mode when dmode=1 raises an illegal instruction exception.

12.1.3 Debug Control and Status Register dcsr

This register gives information about debug capabilities and status. Its detailed functionality is described in the RISC-V Debug Specification 0p13.

12.1.4 Debug PC dpc

When entering Debug Mode, the current PC is copied here. When leaving debug mode, execution resumes at this PC.

12.1.5 Debug Scratch `dscratch`

This register is generally reserved for use by Debug ROM in order to save registers needed by the code in Debug ROM. The debugger may use it as described in the RISC-V Debug Specification 0p13.

12.2 Breakpoints

The U54-MC Core Complex supports 2 hardware breakpoint registers, which can be flexibly shared between debug mode and machine mode.

When a breakpoint register is selected with `tselect`, the other CSRs access the following information for the selected breakpoint:

TDR CSRs when used as Breakpoints		
CSR Name	**Breakpoint Alias**	**Description**
`tselect`	`tselect`	Breakpoint selection index
`tdata1`	`mcontrol`	Breakpoint Match control
`tdata2`	`maddress`	Breakpoint Match address
`tdata3`	N/A	*Reserved*

12.2.1 Breakpoint Match Control Register `mcontrol`

Each breakpoint control register is a read/write register laid out as follows:

Breakpoint Control Register (`mcontrol`)				
Register Offset		`CSR`		
Bits	**Field Name**	**Attr.**	**Rst.**	**Description**
0	R	WARL	X	Address match on LOAD
1	W	WARL	X	Address match on STORE
2	X	WARL	X	Address match on Instruction FETCH
3	U	WARL	X	Address match on User Mode
4	S	WARL	X	Address match on Supervisor Mode
5	H	WARL	X	Address match on Hypervisor Mode
6	M	WARL	X	Address match on Machine Mode
[10:7]	match	WARL	X	Breakpoint Address Match type
11	chain	WARL	0	Chain adjacent conditions.
[17:12]	action	WARL	0	Breakpoint action to take. 0 or 1.
18	timing	WARL	0	Timing of the breakpoint. Always 0.
19	select	WARL	0	Perform match on address or data. Always 0.
20	*Reserved*	WPRI	X	Reserved
[26:21]	maskmax	RO	4	Largest supported NAPOT range
27	dmode	RW	0	Debug-Only access mode
[31:28]	type	RO	2	Address/Data match type, always 2

Table 12.4: Test and Debug Data Register 3

The `type` field is a four-bit read-only field holding the value 2 to indicate this is a breakpoint containing address match logic.

The `bpaction` field is an eight-bit read-write **WARL** field that specifies the available actions when the address match is successful. The value 0 generates a breakpoint exception. The value 1 enters debug mode. Other actions are not implemented.

The R/W/X bits are individual **WARL** fields and if set, indicate an address match should only be successful for loads/stores/instruction fetches respectively, and all combinations of implemented bits must be supported.

The M/H/S/U bits are individual **WARL** fields and if set, indicate that an address match should only be successful in the machine/hypervisor/supervisor/user modes respectively, and all combinations of implemented bits must be supported.

The `match` field is a 4-bit read-write **WARL** field that encodes the type of address range for breakpoint address matching. Three different `match` settings are currently supported: exact, NAPOT, and arbitrary range. A single breakpoint register supports both exact address matches and matches with address ranges that are naturally aligned powers-of-two (NAPOT) in size. Breakpoint registers can be paired to specify arbitrary exact ranges, with the lower-numbered breakpoint register giving the byte address at the bottom of the range and the higher-numbered breakpoint register giving the address one byte above the breakpoint range, and using the `chain` bit to indicate both must match for the action to be taken.

NAPOT ranges make use of low-order bits of the associated breakpoint address register to encode the size of the range as follows:

NAPOT Size Encoding	
maddress	**Match type and size**
`a...aaaaaa`	Exact 1 byte
`a...aaaaa0`	2-byte NAPOT range
`a...aaaa01`	4-byte NAPOT range
`a...aaa011`	8-byte NAPOT range
`a...aa0111`	16-byte NAPOT range
`a...a01111`	32-byte NAPOT range
`...`	...
`a01...1111`	2^{31}-byte NAPOT range

The `maskmax` field is a 6-bit read-only field that specifies the largest supported NAPOT range. The value is the logarithm base 2 of the number of bytes in the largest supported NAPOT range. A value of 0 indicates that only exact address matches are supported (one byte range). A value of 31 corresponds to the maximum NAPOT range, which is 2^{31} bytes in size. The largest range is encoded in `maddress` with the 30 least-signicant bits set to 1, bit 30 set to 0, and bit 31 holding the only address bit considered in the address comparison.

The unary encoding of NAPOT ranges was chosen to reduce the hardware cost of storing and generating the corresponding address mask value.

To provide breakpoints on an exact range, two neighboring breakpoints can be combined with the `chain` bit. The first breakpoint can be set to match on an address using `action` of 2 (greater than or equal). The second breakpoint can be set to match on address using `action of 3` (less than). Setting then `chain` bit on the first breakpoint will then cause it prevent the second breakpoint from firing unless they both match.

12.2.2 Breakpoint Match Address Register (maddress)

Each breakpoint match address register is an XLEN-bit read/write register used to hold significant address bits for address matching, and also the unary-encoded address masking information for NAPOT ranges.

12.2.3 Breakpoint Execution

Breakpoint traps are taken precisely. Implementations that emulate misaligned accesses in software will generate a breakpoint trap when either half of the emulated access falls within the address range. Implementations that support misaligned accesses in hardware must trap if any byte of an access falls within the matching range.

Debug-mode breakpoint traps jump to the debug trap vector without altering machine-mode registers.

Machine-mode breakpoint traps jump to the exception vector with "Breakpoint" set in the mcause register, and with badaddr holding the instruction or data address that caused the trap.

12.2.4 Sharing breakpoints between debug and machine mode

When debug mode uses a breakpoint register, it is no longer visible to machine-mode (i.e., the tdrtype will be 0). Usually, the debugger will grab the breakpoints it needs before entering machine mode, so machine mode will operate with the remaining breakpoint registers.

12.2.5 Sharing breakpoints between debug and machine mode

When debug mode uses a breakpoint register, it is no longer visible to machine-mode (i.e., the tdrtype will be 0). Usually, the debugger will grab the breakpoints it needs before entering machine mode, so machine mode will operate with the remaining breakpoint registers.

12.3 Debug Memory Map

This section describes the debug module's memory map when accessed via the regular system interconnect. The debug module is only accessible to debug code running in debug mode on a hart (or via a debug transport module).

12.3.1 Debug RAM & Program Buffer (0x300–0x3FF)

The U54-MC Core Complex has 16 32-bit words of Program Buffer for the debugger to direct a hart to execute arbitrary RISC-V code. Its location in memory can be determined by executing aiupc instructions and storing the result into the Program Buffer.

The U54-MC Core Complex has 1 32-bit words of Debug Data RAM. Its location can be determined by reading the DMHARTINFO register as described in the RISC-V Debug Specification. This RAM space is used to pass data for the Access Register abstract command described in the RISC-V Debug Specification. The U54-MC Core Complexsupports only GPR register access when harts are halted. All other commands must be implemented by executing from the Debug Program Buffer.

In the U54-MC Core Complex, both the Program Buffer and Debug Data RAM are general purpose RAM and are mapped contiguously in the RISC-V Core IP's memory space. Therefore, additional

data can be passed in the Program Buffer and additional instructions can be stored in the Debug Data RAM.

Debuggers must not execute program buffer programs which access any Debug Module memory except defined Program Buffer and Debug Data addresses.

12.3.2 Debug ROM (0x800–0xFFF)

This ROM region holds the debug routines on SiFive systems. The actual total size may vary between implementations.

12.3.3 Debug Flags (0x100 – 0x110, 0x400 – 0x7FF)

The flag registers in the Debug Module are used for the Debug Module to communicate with each hart. These flags are set and read used by the Debug ROM, and should not be accessed by any program buffer code. The specific behavior of the flags is not further documented here.

12.3.4 Safe Zero Address

In the U54-MC Core Complex, the Debug Module contains the address 0 in the memory map. Reads to this address always return 0, and writes to this address have no impact. This property allows a "safe" location for unprogrammed parts, as the default `mtvec` location is 0x0.

12.4 Instruction Trace Interface

This section describes the interface between a SiFive core and its trace unit. The trace interface conveys information about instruction-retirement and exception events.

Table 12.6 lists the fields in a trace packet. The `valid` signal is 1 if and only if an instruction retires or traps (either by generating a synchronous exception or taking an interrupt). The remaining fields in the packet are only defined when `valid` is 1.

The `iaddr` field holds the address of the instruction that retired or trapped. If address translation is enabled, it is a virtual address, else it is a physical address. Virtual addresses narrower than XLEN bits are sign-extended, and physical addresses narrower than XLEN bits are zero-extended.

The `insn` field holds the instruction that retired or trapped. For instructions narrower than the maximum width, e.g., those in the RISC-V C extension, the unused high-order bits are zero-filled. The length of the instruction can be determined by examining the low-order bits of the instruction, as described in The RISC-V Instruction Set Manual, Volume I: User-Level ISA, Version 2.1 [1]. The width of the `insn` field, ILEN, is 32 bits for current implementations.

The `priv` field indicates the privilege mode at the time of instruction execution. (On an exception, the *next* valid trace packet's `priv` field gives the privilege mode of the activated trap handler.) The width of the `priv` field, PRIVLEN, is 3, and it is encoded as shown in Table 12.5.

The `exception` field is 0 if this packet corresponds to a retired instruction, or 1 if it corresponds to an exception or interrupt. In the former case, the `cause` and `interrupt` fields are undefined and the `tval` field is zero. In the latter case, the fields are set as follows:

- `interrupt` is 0 for synchronous exceptions and 1 for interrupts.
- `cause` supplies the exception or interrupt cause, as would be written to the lower CAUSELEN bits of the `mcause` CSR. For current implementations, CAUSELEN=$\log_2$XLEN.

Value	Description
000	User mode
001	Supervisor mode
011	Machine mode
111	Debug mode

Table 12.5: Encoding of `priv` field in trace interface. Unspecified values are reserved.

- `tval` supplies the associated trap value, e.g., the faulting virtual address for address exceptions, as would be written to the `mtval` CSR.

Future optional extensions may define `tval` to provide ancillary information in cases where it currently supplies zero.

For cores that can retire N instructions per clock cycle, this interface is replicated N times. Lower-numbered entries correspond to older instructions. If fewer than N instructions retire, the valid packets need not be consecutive, i.e., there may be invalid packets between two valid packets. If one of the instructions is an exception, no younger instruction will be valid.

Name	Description
`valid`	Indicates an instruction has retired or trapped.
`iaddr[XLEN-1:0]`	The address of the instruction.
`insn[ILEN-1:0]`	The instruction.
`priv[PRIVLEN-1:0]`	Privilege mode during execution.
`exception`	0 if the instruction retired; 1 if it trapped.
`interrupt`	0 if the exception was synchronous; 1 if interrupt.
`cause[CAUSELEN-1:0]`	Exception cause.
`tval[XLEN-1:0]`	Exception data.

Table 12.6: SiFive instruction trace interface fields.

Chapter 13

Debug Interface

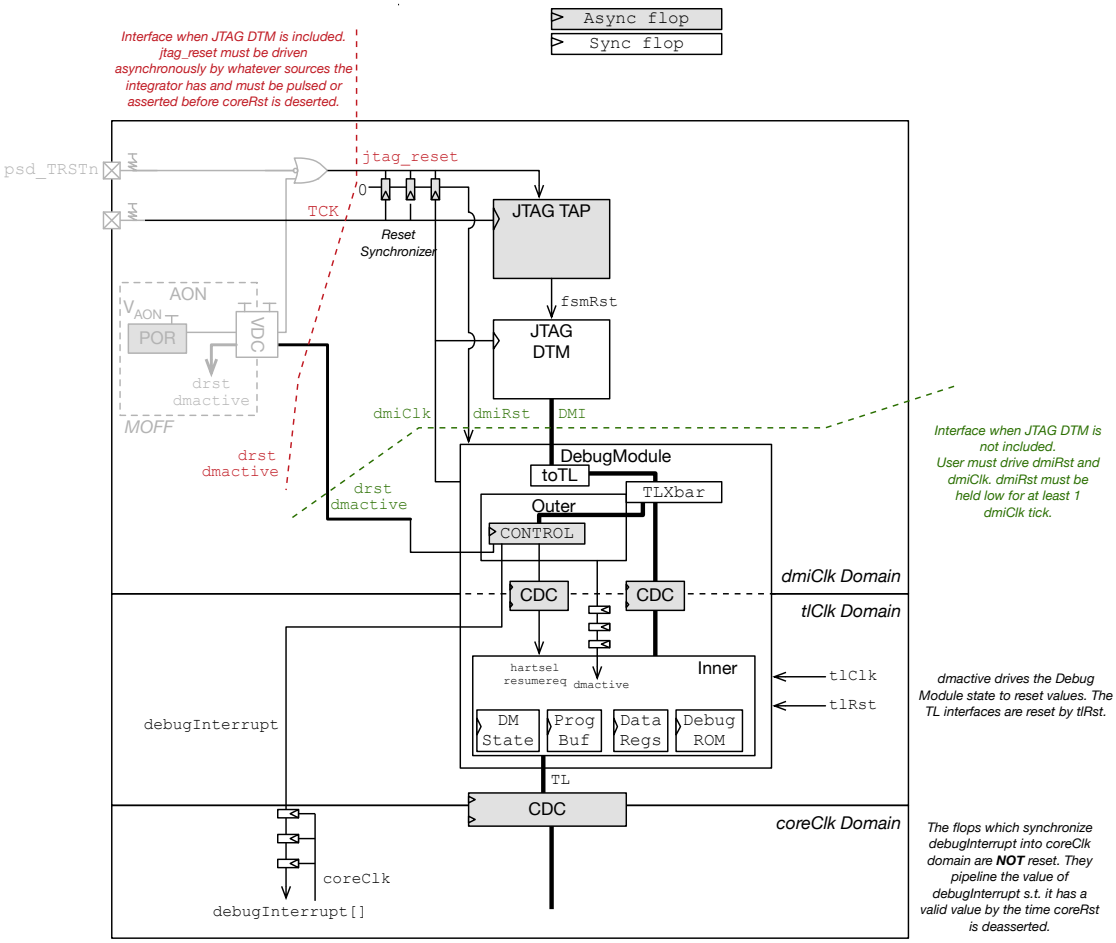

Figure 13.1: Debug Transport Module and Debug Module for HW Debug

The SiFive U54-MC Core Complex includes the JTAG Debug Transport Module described in the RISC-V Debug Specification v0p13. This enables a single external industry-standard 1149.1 JTAG

interface to test and debug the system. The JTAG interface can be directly connected off-chip in a single-chip microcontroller, or can be an embedded JTAG controller for a RISC-V Core IP designed to be included in a larger SoC.

The Debug Transport Module and Debug Module are depicted in Figure 13.1.

On-chip JTAG connections must be driven (no pullups), with a normal two-state driver for TDO under the expectation that on-chip mux logic will be used to select between alternate on-chip JTAG controllers' TDO outputs. TDO logic changes on the falling edge of TCK.

13.1 JTAG TAPC State Machine

The JTAG controller includes the standard TAPC state machine shown in Figure 13.2.

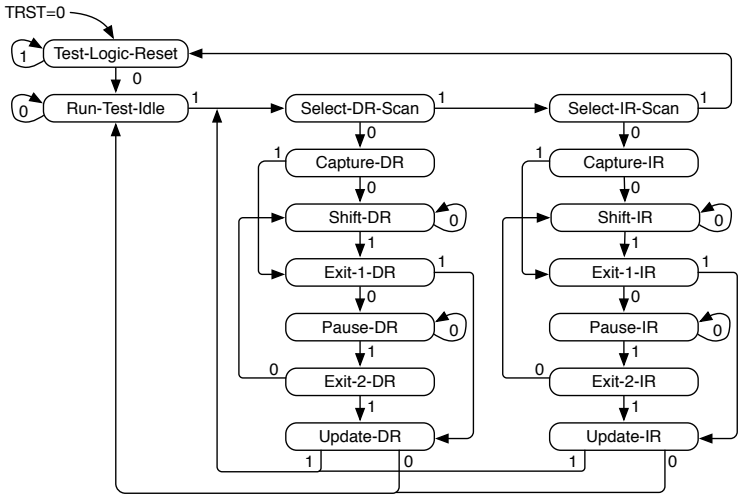

Figure 13.2: JTAG TAPC state machine. The state machine is clocked with TCK. All transitions are labelled with the value on TMS, except for the arc showing asynchronous reset when TRST=0.

13.2 Resetting JTAG logic

The JTAG logic must be asynchrously reset by asserting `jtag_reset` before coreReset is de-asserted.

Asserting `jtag_reset` resets both the JTAG DTM and Debug Module test logic. Because parts of the debug logic require synchronous reset, the `jtag_reset` signal is synchronized inside the U54-MC Core Complex.

During operation the JTAG DTM logic may also be reset without `jtag_reset` by issuing 5 `TCK` clock ticks with `TMS` asserted. This action only resets the JTAG DTM, not the Debug Module.

13.2.1 JTAG Clocking

The JTAG logic always operates in its own clock domain clocked by TCK. The JTAG logic is fully static and has no minimum clock frequency. The maximum TCK frequency is part-specific.

13.2.2 JTAG Standard Instructions

The JTAG DTM implements the BYPASS and IDCODE instructions. The Manufacturer ID field of IDCODE is provided by the RISC-V Core IP integrator, on the `jtag_mfr_id` input.

13.3 JTAG Debug Commands

The JTAG DEBUG instruction gives access to the SiFive debug module by connecting the debug scan register inbetween TDI and TDO.

The debug scan register includes a 2-bit opcode field, a 7-bit debug module address field, and a 32-bit data field to allow various memory-mapped read/write operations to be specified with a single scan of the debug scan register.

These are described in the RISC-V Debug Specification v0p13.

13.4 Using Debug Outputs

The Debug logic in SiFive Systems drives two output signals: `ndreset` and `dmactive`. These signals can be used in integration. It is suggested that the `indreset` signal contribute to the system reset. It must be synchronized before it contributes back to the RISC-V Core IP's overall `reset` signal. This signal must not contribute to the `jtag_reset` signal. The `dmactive` signal may be used to e.g. prevent clock or power gating of the Debug Module logic while debugging is in progress.

Chapter 14

References

Visit the SiFive forums for support and answers to frequently asked questions: `http://forums.sifive.com`.

[1] A. Waterman and K. Asanović, Eds., *The RISC-V Instruction Set Manual, Volume I: User-Level ISA, Version 2.2*, May 2017. [Online]. Available: https://riscv.org/specifications/

[2] ——, *The RISC-V Instruction Set Manual Volume II: Privileged Architecture Version 1.10*, May 2017. [Online]. Available: https://riscv.org/specifications/